PROGRAMACIÓN

LUIS MINGUILLÓN

PROGRAMACION

C#

WINDOWS FORM
EJERCICIOS

Luis Minguillón Pascual

COPYRIGHT

La licencia de este libro electrónico es para uso personal. Por lo tanto no puedes revenderlo a otras personas.

Gracias por respetar los derechos de autor.

©Luis Minguillón Pascual
26/10/2025

Prefacio

La programación es una de las herramientas más poderosas que el ser humano ha desarrollado para transformar ideas en realidades digitales. Este libro nace con la intención de servir como guía práctica para quienes desean aprender y dominar la programación en **C# con Windows Forms**, una de las tecnologías más versátiles y utilizadas en el desarrollo de aplicaciones de escritorio para el sistema operativo Windows.

La programación en **C# Windows Forms** permite crear interfaces gráficas interactivas, intuitivas y funcionales, combinando la potencia del lenguaje C# con la facilidad de uso de un entorno visual. A través de ella, es posible desarrollar programas que integren controles, eventos y componentes visuales que interactúan con el usuario, facilitando el diseño de aplicaciones profesionales, sistemas de gestión, herramientas educativas o software de uso personal.

El propósito de este libro es ofrecer una colección de ejercicios prácticos y progresivos que permitan afianzar los conocimientos en la programación de formularios, eventos, controles y estructuras lógicas. Cada ejercicio está diseñado para estimular el razonamiento lógico, la creatividad y la comprensión del funcionamiento interno de las aplicaciones. De este modo, el lector podrá adquirir experiencia real en el desarrollo de programas completos y funcionales.

Este libro forma parte de una serie dedicada a la práctica y perfeccionamiento de la programación en C#, concebida con el objetivo de acompañar al estudiante desde los fundamentos hasta los proyectos avanzados.

Quiero expresar mi más profundo agradecimiento a mi familia, por su apoyo incondicional, paciencia y confianza durante la creación de esta obra y de toda la serie de libros. Su aliento constante ha sido la fuerza que me ha impulsado a continuar, incluso en los momentos más exigentes del camino.

Con este trabajo espero contribuir a la formación de nuevos programadores, inspirando en ellos la pasión por aprender, crear y explorar el vasto universo del desarrollo de software.

Luis Minguillón Pascual

ÍNDICE

PARTE I — Fundamentos y Lógica

Esta sección se centra en la lógica de programación y el manejo básico de controles de formularios (TextBox, Button, Label).

1. **Calculadora Básica (Suma)**
    - *Objetivo:* Sumar dos números usando `TextBox` y manejar errores de entrada.
2. **Conversor de Temperatura (Celsius a Fahrenheit)**
    - *Objetivo:* Convertir °C a °F (F = C * 9/5 + 32).
3. **Verificador de Número Par/Impar**
    - *Objetivo:* Usar el operador módulo `%` para determinar la paridad de un número entero.
4. **Calculadora de IMC (Índice de Masa Corporal)**
    - *Objetivo:* Calcular IMC y mostrar la categoría de clasificación (Bajo peso, Normal, Sobrepeso, Obesidad).
5. **Contador de Clicks**
    - *Objetivo:* Manejar una variable de instancia (`contadorClicks`) para preservar el estado entre clics.
6. **Mostrar/Ocultar Texto**
    - *Objetivo:* Alternar la propiedad `Visible` de un control.
7. **Cambiar Color de Fondo del Formulario**
    - *Objetivo:* Usar la propiedad `BackColor` y la enumeración `Color` para cambiar el fondo.
8. **Calculadora de Edad (a partir del año)**
    - *Objetivo:* Calcular la edad aproximada restando el año de nacimiento al año actual (`DateTime.Now.Year`).
9. **Generador de Números Aleatorios (1–100)**
    - *Objetivo:* Usar la clase `Random` y el método `Next(minValue, maxValue+1)`.
10. **Conversor de Moneda (Dólares a Euros)**
    - *Objetivo:* Multiplicación simple usando un tipo de cambio fijo.
11. **Calculadora de Área de Rectángulo**
    - *Objetivo:* Calcular Área = base × altura.
12. **Validación de Email Simple (@)**
    - *Objetivo:* Verificar si una cadena contiene el carácter '@' usando el método `Contains`.
13. **Contador de Caracteres**
    - *Objetivo:* Usar la propiedad `.Length` de las cadenas.
14. **Calculadora de Propina (10%)**
    - *Objetivo:* Calcular el 10% de un total y mostrar el resultado en formato monetario (`:C2`).
15. **Reloj Digital (Timer)**
    - *Objetivo:* Mostrar la hora actual en tiempo real utilizando un control `Timer`.
16. **Cambiar Tamaño de Fuente**

- *Objetivo:* Modificar la propiedad `Font` de un control creando un nuevo objeto `Font`.

17. **Calculadora de Promedio (3 números)**
    - *Objetivo:* Calcular el promedio aritmético de tres números.
18. **Verificador de Palíndromo**
    - *Objetivo:* Determinar si un texto se lee igual al derecho y al revés, ignorando espacios y mayúsculas.
19. **Calculadora de Descuento (20%)**
    - *Objetivo:* Aplicar un descuento del 20% (multiplicar por 0.8).
20. **Contador de Vocales**
    - *Objetivo:* Contar cuántas vocales contiene una frase.
21. **Calculadora de Interés Simple**
    - *Objetivo:* Calcular Interés = Capital × Tasa × Tiempo.
22. **Generador de Saludo Personalizado**
    - *Objetivo:* Mostrar un saludo utilizando interpolación de cadenas.
23. **Calculadora de Velocidad**
    - *Objetivo:* Calcular velocidad = distancia / tiempo, controlando que el tiempo sea positivo.
24. **Verificador de Año Bisiesto**
    - *Objetivo:* Usar la condición compuesta de divisibilidad por 4, 100 y 400.
25. **Calculadora de Factorial**
    - *Objetivo:* Calcular el factorial de un número entero positivo usando un bucle `for` y `long` para manejar valores grandes.
26. **Contador de Palabras**
    - *Objetivo:* Contar palabras usando el método `Split` para separar por espacios, tabulaciones y saltos de línea.
27. **Calculadora de Perímetro (Cuadrado)**
    - *Objetivo:* Calcular P = 4 × lado.
28. **Invertir Texto**
    - *Objetivo:* Invertir una cadena usando `Reverse().ToArray()`.
29. **Calculadora de IVA (21%)**
    - *Objetivo:* Calcular el precio final multiplicando por 1.21.
30. **Generador de Contraseña Simple (8 caracteres)**
    - *Objetivo:* Generar una contraseña aleatoria alfanumérica usando `Random` y `Enumerable.Range`.
31. **Calculadora de Hipotenusa**
    - *Objetivo:* Aplicar el Teorema de Pitágoras: `Math.Sqrt(cateto1² + cateto2²)`.
32. **Verificador de Número Primo**
    - *Objetivo:* Verificar divisores solo hasta la raíz cuadrada del número para optimización.
33. **Calculadora de Días entre Fechas**
    - *Objetivo:* Calcular la diferencia en días usando el control `DateTimePicker` y la propiedad `.Days` de `TimeSpan`.
34. **Convertidor de Mayúsculas/Minúsculas**

- *Objetivo:* Usar los métodos `.ToUpper()` y `.ToLower()`.

35. **Calculadora de Volumen de Esfera**
    - *Objetivo:* Aplicar la fórmula $V = 4/3 * \pi * r^3$ usando `Math.PI` y `Math.Pow`.
36. **Contador Regresivo (Timer)**
    - *Objetivo:* Implementar un temporizador que se decrementa en cada evento `timer1_Tick`.
37. **Calculadora de Nota Final (Promedio y Estado)**
    - *Objetivo:* Calcular el promedio de 3 notas y determinar si está "Aprobado" (≥ 5).
38. **Generador de Tabla de Multiplicar**
    - *Objetivo:* Mostrar la tabla del 1 al 10 en un control `ListBox`.
39. **Calculadora de Área de Círculo**
    - *Objetivo:* Calcular $A = \pi * r^2$.
40. **Verificador de Contraseña (Longitud >= 8)**
    - *Objetivo:* Evaluar si la propiedad `.Length` de la contraseña es mayor o igual a 8.
41. **Calculadora de Edad Exacta (Años, Meses, Días)**
    - *Objetivo:* Calcular la edad exacta manejando las diferencias de años, meses y días, y ajustando si son negativos.
42. **Convertidor de Binario a Decimal**
    - *Objetivo:* Usar `Convert.ToInt32` con base 2.
43. **Calculadora de Consumo de Combustible (km/l)**
    - *Objetivo:* Calcular eficiencia (km / litros) controlando la división por cero.
44. **Generador de Números Pares (Primeros 10)**
    - *Objetivo:* Usar un bucle del 1 al 10 multiplicando por 2, y agregando a un `ListBox`.
45. **Calculadora de Salario (Horas × Precio)**
    - *Objetivo:* Calcular el salario multiplicando horas por la tarifa, usando formato monetario.
46. **Verificador de Rango (1 a 100)**
    - *Objetivo:* Verificar la condición lógica `(n >= 1 && n <= 100)`.
47. **Calculadora de Porcentaje**
    - *Objetivo:* Calcular porcentaje (`valor × porcentaje / 100`).
48. **Contador de Números Positivos/Negativos**
    - *Objetivo:* Contar números positivos y negativos a partir de una lista separada por comas.
49. **Calculadora de Tiempo de Viaje (Horas y Minutos)**
    - *Objetivo:* Convertir el tiempo total decimal a horas y minutos separados.
50. **Simulador de Dado (1-6)**
    - *Objetivo:* Simular un lanzamiento usando `Random.Next(1, 7)`.

PARTE II — Manejo de Archivos y Directorios

Esta sección se centra en el uso de las clases `File`, `Directory`, `Path`, `FileInfo` y `DriveInfo` para la gestión del sistema de archivos.

1. **Explorador de Directorios Básico**
    - *Objetivo:* Listar archivos y subdirectorios de una ruta usando `Directory.GetDirectories` y `Directory.GetFiles`.

2. **Verificador de Existencia de Archivo**
    - *Objetivo:* Comprobar la existencia de un archivo con `File.Exists`.
3. **Creador de Directorio**
    - *Objetivo:* Crear una carpeta nueva verificando previamente su existencia con `Directory.CreateDirectory`.
4. **Contador de Archivos en Directorio**
    - *Objetivo:* Contar el número de archivos usando `Directory.GetFiles(ruta).Length`.
5. **Información de Archivo (Tamaño, Extensión, Fechas)**
    - *Objetivo:* Mostrar propiedades de un archivo como tamaño, extensión y fechas, utilizando `FileInfo`.
6. **Copiador de Archivos**
    - *Objetivo:* Copiar un archivo a otra ubicación usando `File.Copy`.
7. **Eliminador de Archivos**
    - *Objetivo:* Eliminar un archivo específico con `File.Delete`.
8. **Renombrador de Archivos**
    - *Objetivo:* Cambiar el nombre de un archivo utilizando `File.Move` y `Path.Combine`.
9. **Buscador de Archivos por Extensión**
    - *Objetivo:* Filtrar archivos por extensión específica usando `Directory.GetFiles(ruta, $"*{ext}")`.
10. **Verificador de Espacio en Disco (DriveInfo)**
    - *Objetivo:* Mostrar el espacio disponible en una unidad utilizando `DriveInfo`.
11. **Creador de Archivo de Texto**
    - *Objetivo:* Crear o sobrescribir un archivo de texto con contenido inicial usando `File.WriteAllText`.
12. **Lector de Archivo de Texto**
    - *Objetivo:* Leer y mostrar el contenido completo de un archivo con `File.ReadAllText`.
13. **Escritor en Archivo de Texto (Sobreescribir)**
    - *Objetivo:* Sobrescribir el contenido de un archivo.
14. **Añadir Texto a Archivo Existente (AppendAllText)**
    - *Objetivo:* Agregar texto al final de un archivo sin borrar el contenido previo usando `File.AppendAllText`.
15. **Listador de Unidades del Sistema**
    - *Objetivo:* Mostrar todas las unidades disponibles con `DriveInfo.GetDrives()`.
16. **Verificador de Tipo de Unidad (Removible, Fija, etc.)**
    - *Objetivo:* Indicar el tipo de unidad usando `DriveInfo.DriveType`.
17. **Contador de Líneas en Archivo de Texto**
    - *Objetivo:* Contar líneas usando `File.ReadAllLines(ruta).Length`.
18. **Buscador de Archivos por Nombre (Contenido del nombre)**
    - *Objetivo:* Listar archivos que contienen una cadena específica en su nombre.
19. **Verificador de Fecha de Modificación**

- *Objetivo:* Mostrar la fecha de última modificación usando `FileInfo.LastWriteTime`.

20. **Creador de Múltiples Directorios**
    - *Objetivo:* Crear varias carpetas a la vez separadas por comas.
21. **Eliminador de Directorio (Vacío)**
    - *Objetivo:* Eliminar un directorio solo si no contiene archivos ni subdirectorios.
22. **Verificador de Archivos Ocultos**
    - *Objetivo:* Detectar si un archivo es oculto usando `FileAttributes.Hidden`.
23. **Calculadora de Tamaño de Directorio (MB)**
    - *Objetivo:* Sumar el tamaño de todos los archivos en un directorio y mostrar el total en MB.
24. **Listador de Extensiones Únicas**
    - *Objetivo:* Mostrar las extensiones sin repetir usando la colección `HashSet<string>`.
25. **Copiador de Directorio (Archivos internos)**
    - *Objetivo:* Copiar todos los archivos de un directorio a otro, creando el destino si es necesario.
26. **Verificador de Archivos de Solo Lectura**
    - *Objetivo:* Detectar el atributo de solo lectura con `FileInfo.IsReadOnly`.
27. **Cambiador de Atributos (Solo Lectura)**
    - *Objetivo:* Establecer o quitar el atributo `IsReadOnly` de un archivo.
28. **Buscador de Archivos Grandes (Tamaño mínimo)**
    - *Objetivo:* Listar archivos que superan un tamaño especificado en MB.
29. **Verificador de Directorio Vacío**
    - *Objetivo:* Determinar si no contiene archivos ni subdirectorios.
30. **Creador de Archivos de Ejemplo**
    - *Objetivo:* Generar múltiples archivos de texto numerados en un directorio.
31. **Comparador de Archivos (Tamaño y Contenido para pequeños)**
    - *Objetivo:* Comparar dos archivos, primero por tamaño y luego byte por byte con `SequenceEqual`.
32. **Listador de Archivos Modificados Hoy**
    - *Objetivo:* Listar archivos cuya fecha de última modificación coincide con `DateTime.Today`.
33. **Verificador de Ruta Válida (Path)**
    - *Objetivo:* Usar `Path.GetFullPath` dentro de un bloque `try/catch` para verificar la validez de la ruta.
34. **Extractor de Información de Ruta**
    - *Objetivo:* Mostrar nombre de archivo, directorio y extensión usando métodos de la clase `Path`.
35. **Creador de Archivo Temporal (Path.GetTempFileName)**
    - *Objetivo:* Generar un archivo único en la carpeta temporal del sistema.
36. **Verificador de Espacio de Archivo (KB, MB, GB)**
    - *Objetivo:* Mostrar el tamaño del archivo en diferentes unidades de medida.
37. **Buscador de Archivos Duplicados (por nombre)**

- *Objetivo:* Encontrar archivos con nombres repetidos utilizando un `Dictionary` para contar las repeticiones.

38. **Organizador por Extensión (Mover a carpetas)**
    - *Objetivo:* Crear carpetas por extensión y mover los archivos a ellas.
39. **Verificador de Archivos del Sistema**
    - *Objetivo:* Identificar archivos protegidos con `FileAttributes.System`.
40. **Creador de Estructura de Directorios Anidada**
    - *Objetivo:* Crear carpetas anidadas automáticamente con una sola llamada a `Directory.CreateDirectory`.
41. **Limpiador de Archivos Temporales (.tmp)**
    - *Objetivo:* Eliminar archivos con extensión `.tmp` usando un filtro en `Directory.GetFiles`.
42. **Verificador de Acceso a Archivo (Escritura)**
    - *Objetivo:* Intentar abrir el archivo en modo escritura para verificar la accesibilidad.
43. **Generador de Lista de Archivos (en un TXT)**
    - *Objetivo:* Guardar la lista de archivos de un directorio en un archivo de texto con `File.WriteAllLines`.
44. **Verificador de Archivos Recientes (Últimos 7 días)**
    - *Objetivo:* Listar archivos modificados desde la fecha límite (`DateTime.Today.AddDays(-7)`).
45. **Creador de Backup Simple (docx, xlsx, pdf)**
    - *Objetivo:* Copiar solo archivos con extensiones específicas a una carpeta de backup.
46. **Verificador de Archivos de Imagen**
    - *Objetivo:* Filtrar archivos con extensiones de imágenes comunes (e.g., .jpg, .png).
47. **Calculadora de Estadísticas de Directorio (Total archivos, tamaño, extensiones)**
    - *Objetivo:* Mostrar la cantidad de archivos, el tamaño total en MB y la lista de extensiones únicas.
48. **Renombrador Masivo (Añadir prefijo)**
    - *Objetivo:* Agregar un prefijo a todos los archivos de un directorio.
49. **Verificador de Archivos de Log**
    - *Objetivo:* Buscar únicamente archivos con extensión `.log`.
50. **Monitor de Cambios en Directorio (FileSystemWatcher)**
    - *Objetivo:* Detectar eventos (crear, modificar, eliminar) en tiempo real en un directorio.

PARTE III — Archivos CSV y JSON

Esta sección cubre la lectura, escritura, validación, conversión y manipulación de archivos CSV (Comma Separated Values) y JSON (JavaScript Object Notation).

1. **Lector Básico de CSV (DataGridView)**
    - *Objetivo:* Cargar un archivo CSV en un control `DataGridView`, usando la primera línea como encabezados.
2. **Generador de CSV Simple**
    - *Objetivo:* Crear un archivo CSV a partir del contenido de un `DataGridView`.
3. **Contador de Filas en CSV (Excluyendo encabezado)**

- *Objetivo:* Contar el total de líneas leídas y restar 1 para excluir la fila de encabezado.

4. **Verificador de Estructura CSV (Columnas consistentes)**
    - *Objetivo:* Comprobar que todas las filas de datos tengan el mismo número de columnas que el encabezado.
5. **Creador de JSON Simple**
    - *Objetivo:* Generar un archivo JSON a partir de un objeto simple usando `JsonSerializer.Serialize`.
6. **Lector de JSON Simple**
    - *Objetivo:* Leer un JSON simple y deserializarlo en un `Dictionary<string, object>` para acceder a sus campos.
7. **Convertidor CSV a JSON**
    - *Objetivo:* Convertir cada fila del CSV en un objeto (diccionario) de una lista y serializarlo a JSON.
8. **Buscador de Texto en CSV**
    - *Objetivo:* Recorrer cada fila del CSV y verificar si contiene el texto buscado.
9. **Validador de JSON (JsonDocument.Parse)**
    - *Objetivo:* Usar `JsonDocument.Parse` dentro de un `try/catch` para verificar si el formato JSON es válido.
10. **Editor de CSV (Cargar y Guardar desde DataGridView)**
    - *Objetivo:* Permitir la carga de un CSV a un `DataGridView` para edición y su posterior guardado (sobrescritura).
11. **Generador de Datos de Prueba CSV (Aleatorios)**
    - *Objetivo:* Crear un CSV con IDs secuenciales, nombres y edades aleatorias.
12. **Extractor de Columnas CSV**
    - *Objetivo:* Localizar una columna por nombre de encabezado y extraer todos sus valores.
13. **Analizador de JSON con Clases (Deserialize)**
    - *Objetivo:* Deserializar JSON en un objeto fuertemente tipado (`public class Persona`).
14. **Filtrador de CSV (Basado en columna/valor)**
    - *Objetivo:* Mostrar las filas que coincidan con un valor exacto en una columna específica.
15. **Creador de JSON Array (Lista de objetos)**
    - *Objetivo:* Generar un JSON que contenga una lista de objetos, donde cada objeto es un elemento del array JSON.
16. **Calculadora de Estadísticas CSV (Suma, promedio, máx.)**
    - *Objetivo:* Convertir los valores de una columna numérica y calcular la suma, el promedio y el valor máximo.
17. **Combinador de Archivos CSV**
    - *Objetivo:* Unir las filas de dos archivos CSV en uno, conservando solo el encabezado del primero.
18. **Editor de JSON Interactivo (Reemplazo simple)**
    - *Objetivo:* Deserializar JSON a `Dictionary`, modificar un valor por su clave y serializarlo de nuevo.
19. **Generador de CSV desde Formulario**

- *Objetivo:* Agregar nuevas filas a un CSV utilizando `StreamWriter`, creando el encabezado si el archivo no existe.

20. **Visualizador de JSON con TreeView (Estructura)**
    - *Objetivo:* Mostrar la estructura jerárquica de un JSON (objetos y arrays) en un control `TreeView` mediante recursión.
21. **Convertidor de JSON a CSV**
    - *Objetivo:* Convertir un JSON (array de objetos) a CSV, usando las claves del primer objeto como encabezados.
22. **Validador de CSV con Encabezados Específicos**
    - *Objetivo:* Comprobar que el CSV contenga todos los encabezados requeridos.
23. **Generador de Configuración JSON**
    - *Objetivo:* Crear un JSON simple utilizado comúnmente para almacenar la configuración de una aplicación.
24. **Lector de Configuración JSON (Deserialización a clase)**
    - *Objetivo:* Leer el archivo de configuración JSON en una clase `Config` para un acceso seguro a las propiedades.
25. **Separador de CSV (Dividir en partes por número de filas)**
    - *Objetivo:* Dividir un CSV grande en varios archivos más pequeños, manteniendo el encabezado en cada parte.
26. **Buscador en JSON (Recursivo en elementos)**
    - *Objetivo:* Buscar un valor específico dentro de un JSON anidado utilizando una función recursiva.
27. **Generador de JSON desde CSV (Con Dictionary)**
    - *Objetivo:* Convertir CSV a una lista de diccionarios que luego se serializa a JSON.
28. **Calculadora de Tamaño de Archivos (Con datos CSV/JSON)**
    - *Objetivo:* Sumar los tamaños de archivos listados en un CSV o JSON que contenga una columna/propiedad "Tamaño".
29. **Comparador de Archivos CSV (Diferencias línea por línea)**
    - *Objetivo:* Comparar dos archivos CSV y mostrar las filas que difieren.
30. **Generador de JSON con Datos Aleatorios para Pruebas**
    - *Objetivo:* Generar un gran volumen de datos de prueba en formato JSON.
31. **Limpiador de CSV (Normalizar espacios y eliminar vacíos)**
    - *Objetivo:* Eliminar espacios en blanco al inicio y al final de las líneas y descartar líneas completamente vacías.
32. **Contador de Elementos JSON (Recursivo)**
    - *Objetivo:* Contar todos los nodos, objetos y valores en un JSON anidado.
33. **Exportador de DataGridView a CSV**
    - *Objetivo:* Guardar el contenido de un `DataGridView` a un archivo CSV.
34. **Creador de JSON Anidado (Estructura compleja)**
    - *Objetivo:* Generar un JSON con objetos dentro de arrays (estructura compleja).
35. **Verificador de Codificación CSV (UTF-8 vs ASCII)**
    - *Objetivo:* Intentar decodificar el archivo en UTF-8 para inferir la codificación.
36. **Modificador de JSON (Búsqueda de camino simple)**
    - *Objetivo:* Modificar un valor en el primer nivel del JSON dado el nombre de la propiedad.

37. **Generador de CSV con Encabezados Personalizados**
    - *Objetivo:* Crear un CSV vacío con encabezados definidos por el usuario (separados por comas).
38. **Validador de Tipos en JSON (Mostrar tipos de datos)**
    - *Objetivo:* Identificar recursivamente el tipo de dato de cada valor en el JSON (String, Number, Array, Object, etc.).
39. **Convertidor de Encoding CSV**
    - *Objetivo:* Leer el archivo en una codificación (e.g., UTF-8) y reescribirlo en otra (e.g., ASCII).
40. **Analizador de Estructura CSV (Mostrar tipos inferidos)**
    - *Objetivo:* Recorrer las columnas e inferir si contienen datos de tipo "Número", "Fecha" o "String".
41. **Generador de JSON Schema (Básico)**
    - *Objetivo:* Crear un esquema JSON para definir los tipos de propiedades y campos obligatorios.
42. **Filtrador de JSON (Búsqueda de propiedad/valor)**
    - *Objetivo:* Filtrar objetos dentro de un JSON array que coincidan con un valor exacto de propiedad.
43. **Calculadora de Hash de Archivos (MD5)**
    - *Objetivo:* Calcular el hash MD5 de un archivo para verificar la integridad.
44. **Generador de Reporte CSV (Estadísticas por columna)**
    - *Objetivo:* Analizar columnas numéricas de un CSV y generar un nuevo CSV con la suma, promedio y máximo.
45. **Comparador de JSON (Identidad de contenido y líneas)**
    - *Objetivo:* Comparar dos archivos JSON línea por línea y mostrar las diferencias.
46. **Normalizador de JSON (Aplicar indentación y CamelCase)**
    - *Objetivo:* Formatear el JSON para que sea legible y aplicar la política de nombres `CamelCase`.
47. **Creador de CSV desde JSON Array (Inversa del Ej. 7)**
    - *Objetivo:* Generar un CSV a partir de un JSON que contiene un array de objetos.
48. **Verificador de Integridad CSV (Campos vacíos y columnas inconsistentes)**
    - *Objetivo:* Detectar filas que tienen un número incorrecto de columnas o contienen campos vacíos.
49. **Generador de Muestra CSV (Primeras N filas)**
    - *Objetivo:* Crear un nuevo CSV con el encabezado y las primeras N filas de datos del archivo original.
50. **Administrador de Archivos CSV/JSON (Resumen de funcionalidades)**
    - *Objetivo:* Aplicación que integra diversas funciones de lectura, filtrado, conversión y generación de reportes.

PARTE I — Fundamentos y Lógica

Ejercicio 1 — Calculadora Básica (Suma)

Objetivo
Crear un formulario que sume dos números introducidos por el usuario y muestre el resultado.

Descripción teórica
Entrada de datos desde TextBox, conversión de `string` a `double`/`decimal`, manejo de errores por entrada no numérica, y evento `Click` en un botón.

Diseño de interfaz (controles y propiedades)

- `TextBox` `txtA` — para el primer número.
- `TextBox` `txtB` — para el segundo número.
- `Button` `btnSumar` — texto: "Sumar".
- `Label` `lblResultado` — texto inicial: "Resultado: ".

Código (Form1.cs — evento Click)

```
private void btnSumar_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtA.Text.Trim(), out double a) &&
        double.TryParse(txtB.Text.Trim(), out double b))
    {
        double resultado = a + b;
        lblResultado.Text = $"Resultado: {resultado}";
    }
    else
    {
        MessageBox.Show("Introduce números válidos.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación paso a paso

1. `TryParse` evita excepciones si la entrada no es numérica.
2. Si ambas conversiones funcionan, suma y muestra en `lblResultado`.
3. Si fallan, muestra un `MessageBox` informando al usuario.

Ejercicio 2 — Conversor de Temperatura (Celsius a Fahrenheit)

Objetivo
Convertir una temperatura en °C a °F usando la fórmula `F = C * 9/5 + 32`.

Descripción teórica
Conversión aritmética simple y validación de entrada.

Diseño de interfaz

- `TextBox` `txtCelsius`.
- `Button` `btnConvertir`.
- `Label` `lblFahrenheit`.

Código

```
private void btnConvertir_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtCelsius.Text.Trim(), out double
celsius))
    {
        double fahrenheit = celsius * 9.0 / 5.0 + 32.0;
        lblFahrenheit.Text = $"°F: {fahrenheit:F2}";
    }
    else
    {
        MessageBox.Show("Introduce un valor numérico para
Celsius.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- Se parsea el valor, se aplica la fórmula y se muestra con 2 decimales (`:F2`).
- Control de errores con `MessageBox`.

Ejercicio 3 — Verificador de Número Par/Impar

Objetivo
Detectar si un número entero es par o impar.

Descripción teórica
Uso del operador módulo `%` para determinar la paridad.

Diseño de interfaz

- `TextBox` `txtNumero`.
- `Button` `btnVerificar`.
- `Label` `lblParidad`.

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtNumero.Text.Trim(), out int n))
    {
        lblParidad.Text = (n % 2 == 0) ? "Par" : "Impar";
    }
    else
    {
        MessageBox.Show("Introduce un entero válido.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- `n % 2 == 0` significa par.
- Uso de `int.TryParse` para seguridad.

Ejercicio 4 — Calculadora de IMC (Índice de Masa Corporal)

Objetivo
Calcular el IMC a partir de peso (kg) y altura (m), y mostrar la categoría.

Descripción teórica
IMC = peso / (altura^2). Clasificaciones básicas (por ejemplo: <18.5 Bajo peso, 18.5–24.9 Normal, 25–29.9 Sobrepeso, ≥30 Obesidad).

Diseño de interfaz

- `TextBox` `txtPeso` (kg).
- `TextBox` `txtAltura` (m).
- `Button` `btnCalcularIMC`.
- `Label` `lblIMC` (valor) y `lblCategoria` (texto).

Código

```
private void btnCalcularIMC_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtPeso.Text.Trim(), out double peso) &&
        double.TryParse(txtAltura.Text.Trim(), out double altura)
&&
        altura > 0)
    {
        double imc = peso / (altura * altura);
        lblIMC.Text = $"IMC: {imc:F2}";

        string categoria;
        if (imc < 18.5) categoria = "Bajo peso";
        else if (imc < 25) categoria = "Normal";
        else if (imc < 30) categoria = "Sobrepeso";
        else categoria = "Obesidad";

        lblCategoria.Text = categoria;
    }
    else
    {
        MessageBox.Show("Introduce peso y altura válidos.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- Verificación de `altura > 0` para evitar división por cero.
- Clasificación por rangos.

Ejercicio 5 — Contador de Clicks

Objetivo
Contar cuántas veces el usuario hace clic en un botón y mostrar el contador.

Descripción teórica
Manejo de variable de instancia que se actualiza en el evento `Click`.

Diseño de interfaz

- `Button` `btnContar` — texto: "Haz clic".
- `Label` `lblClicks` — inicia en "Clicks: 0".

Código (en la clase del formulario)

```
private int contadorClicks = 0;

private void btnContar_Click(object sender, EventArgs e)
{
    contadorClicks++;
    lblClicks.Text = $"Clicks: {contadorClicks}";
}
```

Explicación

- `contadorClicks` es campo del formulario para preservar estado entre clicks.

Ejercicio 6 — Mostrar/Ocultar Texto

Objetivo
Un botón que muestre u oculte un `Label` alternando su visibilidad.

Descripción teórica
Usar la propiedad `Visible` de controles.

Diseño de interfaz

- `Label` `lblSecreto` — mensaje.
- `Button` `btnAlternar` — texto inicial: "Ocultar/Mostrar".

Código

```
private void btnAlternar_Click(object sender, EventArgs e)
{
    lblSecreto.Visible = !lblSecreto.Visible;
    btnAlternar.Text = lblSecreto.Visible ? "Ocultar" : "Mostrar";
}
```

Explicación

- Se invierte `Visible` y se actualiza el texto del botón para reflejar la acción siguiente.

Ejercicio 7 — Cambiar Color de Fondo del Formulario

Objetivo
Permitir cambiar el `BackColor` del formulario mediante botones o un `ComboBox`.

Descripción teórica
Uso de `BackColor` y la enumeración `Color`.

Diseño de interfaz

- `ComboBox` `cmbColores` — items: "Blanco", "Amarillo", "Azul", "Verde".
- `Button` `btnAplicarColor`.

Código

```
private void btnAplicarColor_Click(object sender, EventArgs e)
{
    switch (cmbColores.SelectedItem as string)
    {
        case "Blanco":
            this.BackColor = System.Drawing.Color.White;
            break;
        case "Amarillo":
            this.BackColor = System.Drawing.Color.Yellow;
            break;
        case "Azul":
            this.BackColor = System.Drawing.Color.LightBlue;
            break;
        case "Verde":
            this.BackColor = System.Drawing.Color.LightGreen;
            break;
        default:
            MessageBox.Show("Selecciona un color.", "Info",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            break;
    }
}
```

Explicación

- `this.BackColor` cambia el color del formulario.
- `switch` por opción seleccionada.

Ejercicio 8 — Calculadora de Edad (a partir del año)

Objetivo
Calcular la edad aproximada a partir del año de nacimiento (edad en años, sin meses/días).

Descripción teórica
Obtener el año actual con `DateTime.Now.Year` y restar.

Diseño de interfaz

- `TextBox` `txtAnioNacimiento`.
- `Button` `btnCalcularEdad`.
- `Label` `lblEdad`.

Código

```
private void btnCalcularEdad_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtAnioNacimiento.Text.Trim(), out int anio)
&& anio > 1900 && anio <= DateTime.Now.Year)
    {
        int edad = DateTime.Now.Year - anio;
        lblEdad.Text = $"Edad aproximada: {edad} años";
    }
    else
    {
        MessageBox.Show("Introduce un año de nacimiento válido.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- Validación para años razonables.
- Resta simple para obtener la edad en años completos.

Ejercicio 9 — Generador de Números Aleatorios (1–100)

Objetivo
Generar un número aleatorio entero entre 1 y 100.

Descripción teórica
Uso de `Random` y `Next(minValue, maxValue+1)`.

Diseño de interfaz

- `Button` `btnGenerar`.
- `Label` `lblAleatorio`.

Código

```
private Random rng = new Random();

private void btnGenerar_Click(object sender, EventArgs e)
{
    int num = rng.Next(1, 101); // 1..100 inclusive
    lblAleatorio.Text = $"Número: {num}";
}
```

Explicación

- `Random` debe definirse fuera del evento (campo) para evitar repetir la semilla si se crea en cada click.

Ejercicio 10 — Conversor de Moneda (Dólares a Euros)

Objetivo
Convertir una cantidad en dólares a euros usando un tipo de cambio fijo (por ejemplo, 0.92 EUR por 1 USD). (Nota: para tasas reales habría que actualizar la tasa; aquí usamos valor fijo en el ejercicio.)

Descripción teórica
Multiplicación simple por el tipo de cambio y manejo de formatos numéricos.

Diseño de interfaz

- `TextBox` `txtDolares`.
- `Button` `btnConvertirMoneda`.
- `Label` `lblEuros`.
- (Opcional) `NumericUpDown` `nudTasa` para permitir cambiar la tasa.

Código

```
private const double TASA_DEFAULT = 0.92; // 1 USD = 0.92 EUR

private void btnConvertirMoneda_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtDolares.Text.Trim(), out double
dolares))
    {
        double tasa = TASA_DEFAULT;
        // si usas nudTasa, podrías leer tasa =
(double)nudTasa.Value;
        double euros = dolares * tasa;
        lblEuros.Text = $"€ {euros:F2}";
    }
    else
    {
        MessageBox.Show("Introduce una cantidad en dólares
válida.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- Conversión directa, formato de salida con 2 decimales.
- `TASA_DEFAULT` es constante para el ejercicio; se puede exponer al usuario para variarla.

Ejercicio 11 — Calculadora de Área de Rectángulo

Objetivo
Calcular el área de un rectángulo a partir de su base y altura.

Teoría
El área se calcula con la fórmula: `Área = base × altura`.

Diseño de interfaz

- `TextBox` `txtBase`
- `TextBox` `txtAltura`
- `Button` `btnCalcular`
- `Label` `lblResultado`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtBase.Text, out double baseR) &&
        double.TryParse(txtAltura.Text, out double altura))
    {
        double area = baseR * altura;
        lblResultado.Text = $"Área: {area:F2}";
    }
    else
    {
        MessageBox.Show("Introduce valores numéricos válidos.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- Convierte entradas a números.
- Multiplica base y altura.
- Muestra el resultado con dos decimales.

Ejercicio 12 — Validación de Email Simple (@)

Objetivo
Verificar si un correo electrónico contiene el carácter `'@'`.

Teoría
Uso de métodos de cadena (`Contains`) para validación básica.

Diseño

- `TextBox` `txtEmail`
- `Button` `btnValidar`
- `Label` `lblEstado`

Código

```
private void btnValidar_Click(object sender, EventArgs e)
{
    string email = txtEmail.Text.Trim();
    lblEstado.Text = email.Contains("@") ? "Email válido" : "Email inválido";
}
```

Explicación

- Se evalúa si la cadena incluye `'@'`.
- Se muestra el resultado sin necesidad de expresiones regulares complejas.

Ejercicio 13 — Contador de Caracteres

Objetivo
Contar cuántos caracteres tiene el texto introducido.

Teoría
Propiedad `.Length` de las cadenas.

Diseño

- `TextBox` `txtTexto`
- `Button` `btnContar`
- `Label` `lblCantidad`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    string texto = txtTexto.Text;
    lblCantidad.Text = $"Total de caracteres: {texto.Length}";
}
```

Explicación

- `.Length` devuelve el número de caracteres de la cadena.

Ejercicio 14 — Calculadora de Propina (10%)

Objetivo
Calcular una propina del 10% sobre el total ingresado.

Teoría
`propina = total × 0.10`.

Diseño

- `TextBox` `txtTotal`
- `Button` `btnCalcular`
- `Label` `lblPropina`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtTotal.Text, out double total))
    {
        double propina = total * 0.10;
        lblPropina.Text = $"Propina sugerida: {propina:C2}";
    }
    else
    {
        MessageBox.Show("Introduce un número válido.", "Error");
    }
}
```

Explicación

- El formato `:C2` muestra el valor en formato monetario con 2 decimales.

Ejercicio 15 — Reloj Digital (Timer)

Objetivo
Mostrar la hora actual en tiempo real.

Teoría
Usar un control `Timer` configurado con un intervalo de 1000 ms (1 segundo).

Diseño

- `Label` `lblHora`
- `Timer` `timer1` (Enabled = true, Interval = 1000)

Código

```
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Start();
}

private void timer1_Tick(object sender, EventArgs e)
{
    lblHora.Text = DateTime.Now.ToString("HH:mm:ss");
}
```

Explicación

- `Timer.Tick` se ejecuta cada segundo actualizando la hora.

Ejercicio 16 — Cambiar Tamaño de Fuente

Objetivo
Permitir al usuario cambiar el tamaño de fuente de un texto.

Teoría
Modificar la propiedad `Font` de un control.

Diseño

- `Label` `lblTexto`
- `NumericUpDown` `nudTamaño`
- `Button` `btnCambiar`

Código

```
private void btnCambiar_Click(object sender, EventArgs e)
{
    float tamaño = (float)nudTamaño.Value;
    lblTexto.Font = new Font(lblTexto.Font.FontFamily, tamaño);
}
```

Explicación

- Se crea un nuevo objeto `Font` con el tamaño elegido.
- Se conserva el tipo de letra actual.

Ejercicio 17 — Calculadora de Promedio (3 números)

Objetivo
Calcular el promedio de tres números ingresados.

Teoría
Promedio = (a + b + c) / 3.

Diseño

- `TextBox` `txtA`, `txtB`, `txtC`
- `Button` `btnPromedio`
- `Label` `lblPromedio`

Código

```
private void btnPromedio_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtA.Text, out double a) &&
        double.TryParse(txtB.Text, out double b) &&
        double.TryParse(txtC.Text, out double c))
    {
        double promedio = (a + b + c) / 3;
        lblPromedio.Text = $"Promedio: {promedio:F2}";
    }
    else
    {
        MessageBox.Show("Introduce tres números válidos.");
    }
}
```

Explicación

- Se validan las tres entradas y se calcula el promedio aritmético.

Ejercicio 18 — Verificador de Palíndromo

Objetivo
Determinar si una palabra o frase es un palíndromo.

Teoría
Un texto es palíndromo si se lee igual de izquierda a derecha y viceversa, ignorando espacios y mayúsculas.

Diseño

- `TextBox` `txtPalabra`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string texto = new string(txtPalabra.Text
        .ToLower()
        .Where(c => !char.IsWhiteSpace(c))
        .ToArray());

    string invertido = new string(texto.Reverse().ToArray());

    lblResultado.Text = texto == invertido ? "Es palíndromo" : "No es palíndromo";
}
```

Explicación

- Convierte a minúsculas y elimina espacios.
- Compara la cadena con su versión invertida.

Ejercicio 19 — Calculadora de Descuento (20%)

Objetivo
Aplicar un descuento del 20 % al precio ingresado.

Teoría
`precioFinal = precio - (precio * 0.20)`.

Diseño

- `TextBox` `txtPrecio`
- `Button` `btnCalcular`
- `Label` `lblFinal`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtPrecio.Text, out double precio))
    {
        double final = precio * 0.8;
        lblFinal.Text = $"Precio con descuento: {final:C2}";
    }
    else
    {
        MessageBox.Show("Introduce un precio válido.");
    }
}
```

Explicación

- Multiplicar por `0.8` equivale a descontar el 20 %.
- Se usa formato de moneda.

Ejercicio 20 — Contador de Vocales

Objetivo
Contar cuántas vocales contiene una frase.

Teoría
Recorrer los caracteres y verificar si pertenecen al conjunto de vocales.

Diseño

- `TextBox` `txtFrase`
- `Button` `btnContar`
- `Label` `lblResultado`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    string texto = txtFrase.Text.ToLower();
    int contador = texto.Count(c => "aeiouáéíóú".Contains(c));
    lblResultado.Text = $"Número de vocales: {contador}";
}
```

Explicación

- Convierte a minúsculas para simplificar comparación.
- Cuenta cada carácter que esté en la lista de vocales.

Ejercicio 21 — Calculadora de Interés Simple

Objetivo
Calcular el interés simple: `Interés = Capital × Tasa × Tiempo`.

Teoría
El interés se calcula usando valores decimales y multiplicación directa.

Diseño

- `TextBox` `txtCapital`
- `TextBox` `txtTasa` (en decimal, ej. 0.05 para 5%)
- `TextBox` `txtTiempo` (años)
- `Button` `btnCalcular`
- `Label` `lblInteres`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtCapital.Text, out double capital) &&
        double.TryParse(txtTasa.Text, out double tasa) &&
        double.TryParse(txtTiempo.Text, out double tiempo))
    {
        double interes = capital * tasa * tiempo;
        lblInteres.Text = $"Interés: {interes:C2}";
    }
    else
    {
        MessageBox.Show("Introduce valores válidos.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Explicación

- Validación de entradas.
- Multiplicación directa para obtener el interés.
- Formato monetario para el resultado.

Ejercicio 22 — Generador de Saludo Personalizado

Objetivo
Mostrar un saludo con el nombre ingresado por el usuario.

Teoría
Concatenación de cadenas o interpolación.

Diseño

- `TextBox` `txtNombre`
- `Button` `btnSaludar`
- `Label` `lblSaludo`

Código

```
private void btnSaludar_Click(object sender, EventArgs e)
{
    string nombre = txtNombre.Text.Trim();
    if (!string.IsNullOrEmpty(nombre))
        lblSaludo.Text = $"¡Hola, {nombre}!";
    else
        MessageBox.Show("Introduce tu nombre.", "Error");
}
```

Explicación

- Se recorta el texto y se verifica que no esté vacío.
- Se usa interpolación para formar el mensaje.

Ejercicio 23 — Calculadora de Velocidad

Objetivo
Calcular la velocidad: `velocidad = distancia / tiempo`.

Teoría
Uso de aritmética básica y control de errores (tiempo > 0).

Diseño

- `TextBox` `txtDistancia` (km)
- `TextBox` `txtTiempo` (horas)
- `Button` `btnCalcular`
- `Label` `lblVelocidad`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtDistancia.Text, out double distancia)
&&
        double.TryParse(txtTiempo.Text, out double tiempo) &&
        tiempo > 0)
    {
        double velocidad = distancia / tiempo;
        lblVelocidad.Text = $"Velocidad: {velocidad:F2} km/h";
    }
    else
    {
        MessageBox.Show("Introduce valores válidos y tiempo > 0.",
"Error");
    }
}
```

Explicación

- Validación de entradas y tiempo positivo.
- División simple y formato con dos decimales.

Ejercicio 24 — Verificador de Año Bisiesto

Objetivo
Determinar si un año es bisiesto.

Teoría
Un año es bisiesto si es divisible por 4, pero no por 100, salvo que sea divisible por 400.

Diseño

- `TextBox` `txtAnio`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtAnio.Text, out int anio))
    {
        bool esBisiesto = (anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0);
        lblResultado.Text = esBisiesto ? "Bisiesto" : "No es bisiesto";
    }
    else
    {
        MessageBox.Show("Introduce un año válido.", "Error");
    }
}
```

Explicación

- Condición compuesta para determinar bisiesto.
- Resultado mostrado en `Label`.

Ejercicio 25 — Calculadora de Factorial

Objetivo
Calcular el factorial de un número entero positivo.

Teoría
`n! = n × (n-1) × ... × 1`. Se usa un bucle `for`.

Diseño

- `TextBox` `txtNumero`
- `Button` `btnCalcular`
- `Label` `lblFactorial`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtNumero.Text, out int n) && n >= 0)
    {
        long factorial = 1;
        for (int i = 2; i <= n; i++)
            factorial *= i;

        lblFactorial.Text = $"Factorial: {factorial}";
    }
    else
    {
        MessageBox.Show("Introduce un entero >= 0.", "Error");
    }
}
```

Explicación

- `long` para valores grandes.
- Bucle acumulativo desde 2 hasta n.

Ejercicio 26 — Contador de Palabras

Objetivo
Contar el número de palabras en un texto.

Teoría
Separar por espacios usando `Split` y contar elementos no vacíos.

Diseño

- `TextBox` `txtTexto`
- `Button` `btnContar`
- `Label` `lblPalabras`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    string[] palabras = txtTexto.Text.Split(new char[] { ' ', '\t', '\n' }, StringSplitOptions.RemoveEmptyEntries);
    lblPalabras.Text = $"Número de palabras: {palabras.Length}";
}
```

Explicación

- `Split` con espacios, tab y salto de línea.
- `RemoveEmptyEntries` evita contar espacios extra.

Ejercicio 27 — Calculadora de Perímetro (Cuadrado)

Objetivo
Calcular el perímetro de un cuadrado: `P = 4 × lado`.

Diseño

- `TextBox` `txtLado`
- `Button` `btnCalcular`
- `Label` `lblPerimetro`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtLado.Text, out double lado))
    {
        double perimetro = 4 * lado;
        lblPerimetro.Text = $"Perímetro: {perimetro:F2}";
    }
    else
    {
        MessageBox.Show("Introduce un valor válido.", "Error");
    }
}
```

Explicación

- Multiplicación simple por 4 para perímetro de cuadrado.

Ejercicio 28 — Invertir Texto

Objetivo
Mostrar el texto ingresado invertido.

Teoría
Convertir a arreglo de caracteres, invertir y volver a cadena.

Diseño

- `TextBox` `txtTexto`
- `Button` `btnInvertir`
- `Label` `lblInvertido`

Código

```
private void btnInvertir_Click(object sender, EventArgs e)
{
    string texto = txtTexto.Text;
    string invertido = new string(texto.Reverse().ToArray());
    lblInvertido.Text = invertido;
}
```

Explicación

- `Reverse()` invierte secuencia.
- `ToArray()` y `new string()` generan la cadena final.

Ejercicio 29 — Calculadora de IVA (21%)

Objetivo
Calcular el precio final con IVA 21%.

Diseño

- `TextBox` `txtPrecio`
- `Button` `btnCalcular`
- `Label` `lblFinal`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtPrecio.Text, out double precio))
    {
        double total = precio * 1.21;
        lblFinal.Text = $"Precio con IVA: {total:C2}";
    }
    else
    {
        MessageBox.Show("Introduce un precio válido.", "Error");
    }
}
```

Explicación

- Multiplicar por 1.21 añade el 21% de IVA.
- Formato monetario `:C2`.

Ejercicio 30 — Generador de Contraseña Simple (8 caracteres)

Objetivo
Generar una contraseña aleatoria de 8 caracteres alfanuméricos.

Teoría
Uso de `Random` y un conjunto de caracteres permitidos.

Diseño

- `Button` `btnGenerar`
- `TextBox` `txtContraseña` (solo lectura)

Código

```
private Random rng = new Random();

private const string chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";


private void btnGenerar_Click(object sender, EventArgs e)

{

    string password = new string(Enumerable.Range(0, 8)

        .Select(_ => chars[rng.Next(chars.Length)])

        .ToArray());


    txtContraseña.Text = password;

}
```

Explicación

- `Enumerable.Range` genera índices de 0 a 7.
- `Select` elige un carácter aleatorio.
- Resultado convertido a cadena.

Ejercicio 31 — Calculadora de Hipotenusa

Objetivo
Calcular la hipotenusa de un triángulo rectángulo usando el Teorema de Pitágoras: `hipotenusa = sqrt(cateto1² + cateto2²)`.

Diseño

- `TextBox` `txtCatetoA`
- `TextBox` `txtCatetoB`
- `Button` `btnCalcular`
- `Label` `lblHipotenusa`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtCatetoA.Text, out double a) &&
        double.TryParse(txtCatetoB.Text, out double b))
    {
        double hipotenusa = Math.Sqrt(a * a + b * b);
        lblHipotenusa.Text = $"Hipotenusa: {hipotenusa:F2}";
    }
    else
    {
        MessageBox.Show("Introduce valores numéricos válidos.",
"Error");
    }
}
```

Explicación

- Convierte entradas a `double`.
- Calcula raíz cuadrada de la suma de cuadrados.
- Muestra el resultado con 2 decimales.

Ejercicio 32 — Verificador de Número Primo

Objetivo
Determinar si un número entero es primo.

Diseño

- `TextBox` `txtNumero`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtNumero.Text, out int n) && n > 1)
    {
        bool esPrimo = true;
        for (int i = 2; i <= Math.Sqrt(n); i++)
        {
            if (n % i == 0)
            {
                esPrimo = false;
                break;
            }
        }
        lblResultado.Text = esPrimo ? "Número primo" : "No es
primo";
    }
    else
    {
        MessageBox.Show("Introduce un entero mayor que 1.",
"Error");
    }
}
```

Explicación

- Verifica divisores hasta la raíz cuadrada del número para optimizar.
- Muestra resultado en `Label`.

Ejercicio 33 — Calculadora de Días entre Fechas

Objetivo
Calcular la diferencia en días entre dos fechas.

Diseño

- `DateTimePicker` `dtpInicio`
- `DateTimePicker` `dtpFin`
- `Button` `btnCalcular`
- `Label` `lblDias`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    DateTime inicio = dtpInicio.Value.Date;
    DateTime fin = dtpFin.Value.Date;
    int dias = (fin - inicio).Days;
    lblDias.Text = $"Días de diferencia: {dias}";
}
```

Explicación

- `DateTimePicker.Value` obtiene fecha seleccionada.
- La diferencia entre fechas es un `TimeSpan`; `.Days` devuelve los días completos.

Ejercicio 34 — Convertidor de Mayúsculas/Minúsculas

Objetivo
Convertir un texto a mayúsculas o minúsculas según botón.

Diseño

- `TextBox` `txtTexto`
- `Button` `btnMayusculas`
- `Button` `btnMinusculas`
- `Label` `lblResultado`

Código

```
private void btnMayusculas_Click(object sender, EventArgs e)
{
    lblResultado.Text = txtTexto.Text.ToUpper();
}

private void btnMinusculas_Click(object sender, EventArgs e)
{
    lblResultado.Text = txtTexto.Text.ToLower();
}
```

Explicación

- `.ToUpper()` y `.ToLower()` convierten el texto.
- Resultado se muestra en `Label`.

Ejercicio 35 — Calculadora de Volumen de Esfera

Objetivo
Calcular el volumen de una esfera: `V = 4/3 * π * r³`.

Diseño

- `TextBox` `txtRadio`
- `Button` `btnCalcular`
- `Label` `lblVolumen`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtRadio.Text, out double r) && r >= 0)
    {
        double volumen = (4.0 / 3.0) * Math.PI * Math.Pow(r, 3);
        lblVolumen.Text = $"Volumen: {volumen:F2}";
    }
    else
    {
        MessageBox.Show("Introduce un radio válido.", "Error");
    }
}
```

Explicación

- `Math.Pow(r,3)` eleva al cubo.
- `Math.PI` proporciona el valor de π.

Ejercicio 36 — Contador Regresivo (Timer)

Objetivo
Hacer un temporizador que cuente regresivamente.

Diseño

- `TextBox` `txtSegundos`
- `Button` `btnIniciar`
- `Label` `lblCuenta`
- `Timer` `timer1` (Interval = 1000 ms)

Código

```
private int segundos;

private void btnIniciar_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtSegundos.Text, out segundos) && segundos > 0)
    {
        lblCuenta.Text = segundos.ToString();
        timer1.Start();
    }
    else
    {
        MessageBox.Show("Introduce un número positivo.", "Error");
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    if (segundos > 0)
    {
        segundos--;
        lblCuenta.Text = segundos.ToString();
    }
```

```
    else
    {
        timer1.Stop();
        MessageBox.Show("Tiempo finalizado", "Aviso");
    }
}
```

Explicación

- Cada tick del timer decrementa el contador y actualiza el `Label`.
- Se detiene y muestra aviso al llegar a cero.

Ejercicio 37 — Calculadora de Nota Final (Promedio y Estado)

Objetivo
Calcular promedio de 3 notas y mostrar estado: "Aprobado" ≥ 5.

Diseño

- `TextBox` `txtNota1`, `txtNota2`, `txtNota3`
- `Button` `btnCalcular`
- `Label` `lblPromedio`, `lblEstado`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtNota1.Text, out double n1) &&
        double.TryParse(txtNota2.Text, out double n2) &&
        double.TryParse(txtNota3.Text, out double n3))
    {
        double promedio = (n1 + n2 + n3) / 3;
        lblPromedio.Text = $"Promedio: {promedio:F2}";
        lblEstado.Text = promedio >= 5 ? "Aprobado" : "Suspenso";
    }
    else
    {
        MessageBox.Show("Introduce notas válidas.", "Error");
    }
}
```

Explicación

- Calcula promedio y evalúa condición de aprobado/suspenso.

Ejercicio 38 — Generador de Tabla de Multiplicar

Objetivo
Mostrar tabla de multiplicar de un número del 1 al 10.

Diseño

- `TextBox` `txtNumero`
- `Button` `btnGenerar`
- `ListBox` `lstTabla`

Código

```
private void btnGenerar_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtNumero.Text, out int n))
    {
        lstTabla.Items.Clear();
        for (int i = 1; i <= 10; i++)
        {
            lstTabla.Items.Add($"{n} x {i} = {n * i}");
        }
    }
    else
    {
        MessageBox.Show("Introduce un número válido.", "Error");
    }
}
```

Explicación

- Limpia `ListBox` y agrega línea por línea de la tabla de multiplicar.

Ejercicio 39 — Calculadora de Área de Círculo

Objetivo
Calcular área de círculo: `A = π * r²`.

Diseño

- `TextBox` `txtRadio`
- `Button` `btnCalcular`
- `Label` `lblArea`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtRadio.Text, out double r) && r >= 0)
    {
        double area = Math.PI * r * r;
        lblArea.Text = $"Área: {area:F2}";
    }
    else
    {
        MessageBox.Show("Introduce un radio válido.", "Error");
    }
}
```

Explicación

- Eleva al cuadrado el radio y multiplica por π.

Ejercicio 40 — Verificador de Contraseña (Longitud >= 8)

Objetivo
Validar que una contraseña tenga al menos 8 caracteres.

Diseño

- `TextBox` `txtPassword` (PasswordChar = '*')
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string pwd = txtPassword.Text;
    lblResultado.Text = pwd.Length >= 8 ? "Contraseña válida" :
"Contraseña inválida";
}
```

Explicación

- Evalúa longitud mínima y muestra resultado en `Label`.

Ejercicio 41 — Calculadora de Edad Exacta (Años, Meses, Días)

Objetivo
Calcular la edad exacta de una persona en años, meses y días.

Diseño

- `DateTimePicker` `dtpNacimiento`
- `Button` `btnCalcular`
- `Label` `lblEdad`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    DateTime nacimiento = dtpNacimiento.Value;
    DateTime hoy = DateTime.Today;

    int años = hoy.Year - nacimiento.Year;
    int meses = hoy.Month - nacimiento.Month;
    int dias = hoy.Day - nacimiento.Day;

    if (dias < 0)
    {
        meses--;
        dias += DateTime.DaysInMonth(hoy.Year, hoy.Month - 1);
    }
    if (meses < 0)
    {
        años--;
        meses += 12;
    }

    lblEdad.Text = $"Edad: {años} años, {meses} meses, {dias}
días";
}
```

Explicación

- Calcula diferencia año/mes/día.
- Ajusta si día o mes son negativos.
- Muestra edad exacta.

Ejercicio 42 — Convertidor de Binario a Decimal

Objetivo
Convertir un número binario (cadena) a decimal.

Diseño

- `TextBox` `txtBinario`
- `Button` `btnConvertir`
- `Label` `lblDecimal`

Código

```
private void btnConvertir_Click(object sender, EventArgs e)
{
    string bin = txtBinario.Text.Trim();
    try
    {
        int decimalValue = Convert.ToInt32(bin, 2);
        lblDecimal.Text = $"Decimal: {decimalValue}";
    }
    catch
    {
        MessageBox.Show("Introduce un binario válido (0 y 1).",
"Error");
    }
}
```

Explicación

- `Convert.ToInt32` con base 2 convierte binario a decimal.
- Captura excepciones para entradas inválidas.

Ejercicio 43 — Calculadora de Consumo de Combustible (km/l)

Objetivo
Calcular eficiencia de combustible: `consumo = km / litros`.

Diseño

- `TextBox` `txtKm`
- `TextBox` `txtLitros`
- `Button` `btnCalcular`
- `Label` `lblConsumo`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtKm.Text, out double km) &&
        double.TryParse(txtLitros.Text, out double litros) &&
litros > 0)
    {
        double consumo = km / litros;
        lblConsumo.Text = $"Consumo: {consumo:F2} km/l";
    }
    else
    {
        MessageBox.Show("Introduce valores válidos y litros > 0.",
"Error");
    }
}
```

Explicación

- División simple y control de litros > 0 para evitar errores.

Ejercicio 44 — Generador de Números Pares (Primeros 10)

Objetivo
Generar y mostrar los primeros 10 números pares.

Diseño

- `Button` `btnGenerar`
- `ListBox` `lstPares`

Código

```
private void btnGenerar_Click(object sender, EventArgs e)
{
    lstPares.Items.Clear();
    for (int i = 1; i <= 10; i++)
    {
        lstPares.Items.Add(i * 2);
    }
}
```

Explicación

- Bucle del 1 al 10 multiplicando por 2.
- Resultados agregados a `ListBox`.

Ejercicio 45 — Calculadora de Salario (Horas × Precio)

Objetivo
Calcular salario según horas trabajadas y tarifa por hora.

Diseño

- `TextBox` `txtHoras`
- `TextBox` `txtPrecioHora`
- `Button` `btnCalcular`
- `Label` `lblSalario`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtHoras.Text, out double horas) &&
        double.TryParse(txtPrecioHora.Text, out double precio))
    {
        double salario = horas * precio;
        lblSalario.Text = $"Salario: {salario:C2}";
    }
    else
    {
        MessageBox.Show("Introduce valores válidos.", "Error");
    }
}
```

Explicación

- Multiplicación directa y formato monetario.

Ejercicio 46 — Verificador de Rango (1 a 100)

Objetivo
Verificar si un número está entre 1 y 100.

Diseño

- `TextBox` `txtNumero`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtNumero.Text, out int n))
    {
        lblResultado.Text = (n >= 1 && n <= 100) ? "Dentro del rango" : "Fuera del rango";
    }
    else
    {
        MessageBox.Show("Introduce un número válido.", "Error");
    }
}
```

Explicación

- Condición lógica simple para rango 1–100.

Ejercicio 47 — Calculadora de Porcentaje

Objetivo
Calcular el porcentaje de un número: `resultado = valor × porcentaje / 100`.

Diseño

- `TextBox` `txtValor`
- `TextBox` `txtPorcentaje`
- `Button` `btnCalcular`
- `Label` `lblResultado`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtValor.Text, out double valor) &&
        double.TryParse(txtPorcentaje.Text, out double
porcentaje))
    {
        double resultado = valor * porcentaje / 100;
        lblResultado.Text = $"Resultado: {resultado:F2}";
    }
    else
    {
        MessageBox.Show("Introduce valores válidos.", "Error");
    }
}
```

Explicación

- Multiplicación y división por 100 para obtener porcentaje.

Ejercicio 48 — Contador de Números Positivos/Negativos

Objetivo
Contar cuántos números positivos y negativos se ingresan.

Diseño

- `TextBox` `txtNumeros` (números separados por comas)
- `Button` `btnContar`
- `Label` `lblPositivos`, `lblNegativos`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    string[] numeros = txtNumeros.Text.Split(',');
    int positivos = 0, negativos = 0;

    foreach (var num in numeros)
    {
        if (double.TryParse(num.Trim(), out double n))
        {
            if (n > 0) positivos++;
            else if (n < 0) negativos++;
        }
    }

    lblPositivos.Text = $"Positivos: {positivos}";
    lblNegativos.Text = $"Negativos: {negativos}";
}
```

Explicación

- Separa números por comas, convierte y clasifica según signo.

Ejercicio 49 — Calculadora de Tiempo de Viaje (Horas y Minutos)

Objetivo
Calcular duración del viaje en horas y minutos.

Diseño

- `TextBox` `txtDistancia` (km)
- `TextBox` `txtVelocidad` (km/h)
- `Button` `btnCalcular`
- `Label` `lblTiempo`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    if (double.TryParse(txtDistancia.Text, out double distancia)
&&
        double.TryParse(txtVelocidad.Text, out double velocidad)
&& velocidad > 0)
    {
        double tiempoTotal = distancia / velocidad;
        int horas = (int)tiempoTotal;
        int minutos = (int)((tiempoTotal - horas) * 60);
        lblTiempo.Text = $"Tiempo estimado: {horas}h {minutos}m";
    }
    else
    {
        MessageBox.Show("Introduce valores válidos y velocidad >
0.", "Error");
    }
}
```

Explicación

- Convierte tiempo decimal a horas y minutos.
- Control de velocidad positiva para evitar división por cero.

Ejercicio 50 — Simulador de Dado (1-6)

Objetivo
Simular lanzamiento de un dado de 6 caras.

Diseño

- `Button` `btnLanzar`
- `Label` `lblDado`
- `Random rng = new Random()`

Código

```
private Random rng = new Random();

private void btnLanzar_Click(object sender, EventArgs e)
{
    int dado = rng.Next(1, 7); // 1 a 6 inclusive
    lblDado.Text = $"Resultado: {dado}";
}
```

Explicación

- `Random.Next(1,7)` genera números del 1 al 6.
- Muestra resultado en `Label`.

PARTE II — Manejo de Archivos y Directorios

Ejercicio 1 — Explorador de Directorios Básico

Objetivo
Listar todos los archivos y subdirectorios de una carpeta.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnExplorar`
- `ListBox` `lstArchivos`

Código

```
private void btnExplorar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        lstArchivos.Items.Clear();
        foreach (var dir in Directory.GetDirectories(ruta))
            lstArchivos.Items.Add("[DIR] " + dir);
        foreach (var file in Directory.GetFiles(ruta))
            lstArchivos.Items.Add(file);
    }
    else
    {
        MessageBox.Show("Ruta no válida.", "Error");
    }
}
```

Explicación

- `Directory.GetDirectories` y `Directory.GetFiles` listan contenido.
- Se diferencian carpetas de archivos con prefijo `[DIR]`.

Ejercicio 2 — Verificador de Existencia de Archivo

Objetivo
Comprobar si un archivo existe.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    lblResultado.Text = File.Exists(ruta) ? "Archivo existe" :
"Archivo no encontrado";
}
```

Explicación

- `File.Exists` devuelve `true` si el archivo existe, `false` si no.

Ejercicio 3 — Creador de Directorio

Objetivo
Crear una carpeta nueva en la ruta especificada.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnCrear`
- `Label` `lblResultado`

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (!Directory.Exists(ruta))
    {
        Directory.CreateDirectory(ruta);
        lblResultado.Text = "Directorio creado";
    }
    else
    {
        lblResultado.Text = "El directorio ya existe";
    }
}
```

Explicación

- Se verifica existencia antes de crear con `Directory.CreateDirectory`.

Ejercicio 4 — Contador de Archivos en Directorio

Objetivo
Contar cuántos archivos hay en una carpeta.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnContar`
- `Label` `lblCantidad`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        int cantidad = Directory.GetFiles(ruta).Length;
        lblCantidad.Text = $"Archivos: {cantidad}";
    }
    else
    {
        MessageBox.Show("Ruta no válida.", "Error");
    }
}
```

Explicación

- `Directory.GetFiles(ruta).Length` devuelve número de archivos.

Ejercicio 5 — Información de Archivo (Tamaño, Extensión, Fechas)

Objetivo
Mostrar información básica de un archivo.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnInfo`
- `Label` `lblInfo`

Código

```
private void btnInfo_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        FileInfo fi = new FileInfo(ruta);
        lblInfo.Text = $"Tamaño: {fi.Length} bytes\nExtensión:
{fi.Extension}\nCreado: {fi.CreationTime}\nModificado:
{fi.LastWriteTime}";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `FileInfo` permite acceder a propiedades como tamaño, extensión y fechas.

Ejercicio 6 — Copiador de Archivos

Objetivo
Copiar un archivo a otra ubicación.

Diseño

- `TextBox` `txtOrigen`
- `TextBox` `txtDestino`
- `Button` `btnCopiar`

Código

```
private void btnCopiar_Click(object sender, EventArgs e)
{
    string origen = txtOrigen.Text;
    string destino = txtDestino.Text;
    if (File.Exists(origen))
    {
        File.Copy(origen, destino, true);
        MessageBox.Show("Archivo copiado.", "Éxito");
    }
    else
    {
        MessageBox.Show("Archivo de origen no encontrado.",
"Error");
    }
}
```

Explicación

- `File.Copy` copia archivo; el `true` permite sobrescribir si existe.

Ejercicio 7 — Eliminador de Archivos

Objetivo
Eliminar un archivo específico.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnEliminar`

Código

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        File.Delete(ruta);
        MessageBox.Show("Archivo eliminado.", "Éxito");
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `File.Delete` elimina el archivo indicado.

Ejercicio 8 — Renombrador de Archivos

Objetivo
Cambiar el nombre de un archivo.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtNuevoNombre`
- `Button` `btnRenombrar`

Código

```
private void btnRenombrar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (File.Exists(ruta))
    {
        string directorio = Path.GetDirectoryName(ruta);
        string nuevo = Path.Combine(directorio,
txtNuevoNombre.Text);
        File.Move(ruta, nuevo);
        MessageBox.Show("Archivo renombrado.", "Éxito");
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `Path.GetDirectoryName` obtiene carpeta.
- `File.Move` permite renombrar.

Ejercicio 9 — Buscador de Archivos por Extensión

Objetivo
Listar todos los archivos de una extensión específica.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtExtension`
- `Button` `btnBuscar`
- `ListBox` `lstArchivos`

Código

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string ext = txtExtension.Text;
    if (Directory.Exists(ruta))
    {
        lstArchivos.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta, $"*{ext}"))
            lstArchivos.Items.Add(file);
    }
    else
    {
        MessageBox.Show("Ruta no válida.", "Error");
    }
}
```

Explicación

- `Directory.GetFiles(ruta, "*ext")` filtra archivos por extensión.

Ejercicio 10 — Verificador de Espacio en Disco (DriveInfo)

Objetivo
Mostrar espacio disponible en una unidad.

Diseño

- `TextBox` `txtUnidad` (ej. "C:\")
- `Button` `btnVerificar`
- `Label` `lblEspacio`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    try
    {
        DriveInfo drive = new DriveInfo(txtUnidad.Text);
        lblEspacio.Text = $"Disponible:
{drive.AvailableFreeSpace / (1024*1024*1024)} GB";
    }
    catch
    {
        MessageBox.Show("Unidad no válida.", "Error");
    }
}
```

Explicación

- `DriveInfo` proporciona información de espacio libre.
- Convierte bytes a GB para mostrar.

Ejercicio 11 — Creador de Archivo de Texto

Objetivo
Crear un archivo de texto vacío o con contenido inicial.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtContenido`
- `Button` `btnCrear`

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string contenido = txtContenido.Text;
    File.WriteAllText(ruta, contenido);
    MessageBox.Show("Archivo creado.", "Éxito");
}
```

Explicación

- `File.WriteAllText` crea o sobrescribe archivo con contenido dado.

Ejercicio 12 — Lector de Archivo de Texto

Objetivo
Leer y mostrar el contenido de un archivo de texto.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnLeer`
- `TextBox` `txtContenido` (Multilínea = true)

Código

```
private void btnLeer_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (File.Exists(ruta))
    {
        txtContenido.Text = File.ReadAllText(ruta);
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `File.ReadAllText` devuelve todo el contenido como string.

Ejercicio 13 — Escritor en Archivo de Texto (Sobreescribir)

Objetivo
Sobrescribir contenido de un archivo existente.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtContenido`
- `Button` `btnSobrescribir`

Código

```
private void btnSobrescribir_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string contenido = txtContenido.Text;
    File.WriteAllText(ruta, contenido);
    MessageBox.Show("Archivo sobrescrito.", "Éxito");
}
```

Explicación

- Igual que el ejercicio 11, sobrescribe el archivo existente.

Ejercicio 14 — Añadir Texto a Archivo Existente (AppendAllText)

Objetivo
Agregar texto al final de un archivo sin borrar contenido previo.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtContenido`
- `Button` `btnAgregar`

Código

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string contenido = txtContenido.Text;
    File.AppendAllText(ruta, contenido + Environment.NewLine);
    MessageBox.Show("Texto agregado.", "Éxito");
}
```

Explicación

- `File.AppendAllText` agrega contenido al final del archivo.

Ejercicio 15 — Listador de Unidades del Sistema

Objetivo
Mostrar todas las unidades disponibles en el sistema.

Diseño

- `Button` `btnListar`
- `ListBox` `lstUnidades`

Código

```
private void btnListar_Click(object sender, EventArgs e)
{
    lstUnidades.Items.Clear();
    foreach (var drive in DriveInfo.GetDrives())
    {
        lstUnidades.Items.Add($"{drive.Name} - Tipo: {drive.DriveType}");
    }
}
```

Explicación

- `DriveInfo.GetDrives()` devuelve todas las unidades.
- Se muestra nombre y tipo (Removible, Fijo, etc.).

Ejercicio 16 — Verificador de Tipo de Unidad (Removible, Fija, etc.)

Objetivo
Indicar el tipo de unidad de una letra de disco.

Diseño

- `TextBox` `txtUnidad`
- `Button` `btnVerificar`
- `Label` `lblTipo`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    try
    {
        DriveInfo drive = new DriveInfo(txtUnidad.Text);
        lblTipo.Text = $"Tipo: {drive.DriveType}";
    }
    catch
    {
        MessageBox.Show("Unidad no válida.", "Error");
    }
}
```

Explicación

- `DriveInfo.DriveType` devuelve `Fixed`, `Removable`, `CDRom`, etc.

Ejercicio 17 — Contador de Líneas en Archivo de Texto

Objetivo
Contar cuántas líneas tiene un archivo de texto.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnContar`
- `Label` `lblLineas`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        int lineas = File.ReadAllLines(ruta).Length;
        lblLineas.Text = $"Líneas: {lineas}";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `File.ReadAllLines` devuelve array de líneas; `.Length` indica cantidad.

Ejercicio 18 — Buscador de Archivos por Nombre (Contenido del nombre)

Objetivo
Listar archivos que contienen una cadena en su nombre.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtBusqueda`
- `Button` `btnBuscar`
- `ListBox` `lstArchivos`

Código

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string busqueda = txtBusqueda.Text;
    if (Directory.Exists(ruta))
    {
        lstArchivos.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta))
        {
            if (Path.GetFileName(file).Contains(busqueda))
                lstArchivos.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Ruta no válida.", "Error");
    }
}
```

Explicación

- Filtra archivos que contienen la palabra indicada en su nombre.

Ejercicio 19 — Verificador de Fecha de Modificación

Objetivo
Mostrar la fecha de la última modificación de un archivo.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnFecha`
- `Label` `lblFecha`

Código

```
private void btnFecha_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        FileInfo fi = new FileInfo(ruta);
        lblFecha.Text = $"Última modificación:
{fi.LastWriteTime}";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `FileInfo.LastWriteTime` devuelve la fecha de última modificación.

Ejercicio 20 — Creador de Múltiples Directorios

Objetivo
Crear varias carpetas a la vez.

Diseño

- `TextBox` `txtRutaBase`
- `TextBox` `txtCarpetas` (Separadas por comas)
- `Button` `btnCrear`

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string rutaBase = txtRutaBase.Text;
    string[] carpetas = txtCarpetas.Text.Split(',');

    foreach (var carpeta in carpetas)
    {
        string ruta = Path.Combine(rutaBase, carpeta.Trim());
        if (!Directory.Exists(ruta))
            Directory.CreateDirectory(ruta);
    }

    MessageBox.Show("Directorios creados.", "Éxito");
}
```

Explicación

- Se crean múltiples carpetas usando `Directory.CreateDirectory`.
- `Path.Combine` asegura la ruta correcta.

Ejercicio 21 — Eliminador de Directorio (Vacío)

Objetivo
Eliminar un directorio solo si está vacío.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnEliminar`

Código

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        if (Directory.GetFiles(ruta).Length == 0 &&
Directory.GetDirectories(ruta).Length == 0)
        {
            Directory.Delete(ruta);
            MessageBox.Show("Directorio eliminado.", "Éxito");
        }
        else
        {
            MessageBox.Show("El directorio no está vacío.",
"Error");
        }
    }
    else
    {
        MessageBox.Show("Directorio no encontrado.", "Error");
    }
}
```

Explicación

- Comprueba si el directorio tiene archivos o subdirectorios antes de eliminar.

Ejercicio 22 — Verificador de Archivos Ocultos

Objetivo
Verificar si un archivo es oculto.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        FileInfo fi = new FileInfo(ruta);
        lblResultado.Text =
fi.Attributes.HasFlag(FileAttributes.Hidden) ? "Archivo oculto" :
"Archivo visible";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `FileAttributes.Hidden` indica si el archivo está oculto.

Ejercicio 23 — Calculadora de Tamaño de Directorio (MB)

Objetivo
Calcular el tamaño total de todos los archivos en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnCalcular`
- `Label` `lblTamaño`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        long tamaño = 0;
        foreach (var file in Directory.GetFiles(ruta))
        {
            FileInfo fi = new FileInfo(file);
            tamaño += fi.Length;
        }
        lblTamaño.Text = $"Tamaño: {tamaño / (1024 * 1024)} MB";
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Suma tamaño de todos los archivos y convierte bytes a MB.

Ejercicio 24 — Listador de Extensiones Únicas

Objetivo
Mostrar las extensiones únicas de archivos en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnListar`
- `ListBox` `lstExtensiones`

Código

```
private void btnListar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        var extensiones = new HashSet<string>();
        foreach (var file in Directory.GetFiles(ruta))
            extensiones.Add(Path.GetExtension(file));

        lstExtensiones.Items.Clear();
        foreach (var ext in extensiones)
            lstExtensiones.Items.Add(ext);
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- `HashSet` asegura que no se repitan extensiones.

Ejercicio 25 — Copiador de Directorio (Archivos internos)

Objetivo
Copiar todos los archivos de un directorio a otro.

Diseño

- `TextBox` `txtOrigen`
- `TextBox` `txtDestino`
- `Button` `btnCopiar`

Código

```
private void btnCopiar_Click(object sender, EventArgs e)
{
    string origen = txtOrigen.Text;
    string destino = txtDestino.Text;
    if (Directory.Exists(origen))
    {
        Directory.CreateDirectory(destino);
        foreach (var file in Directory.GetFiles(origen))
        {
            string nombre = Path.GetFileName(file);
            File.Copy(file, Path.Combine(destino, nombre), true);
        }
        MessageBox.Show("Archivos copiados.", "Éxito");
    }
    else
    {
        MessageBox.Show("Directorio origen no válido.", "Error");
    }
}
```

Explicación

- Copia cada archivo individualmente; crea destino si no existe.

Ejercicio 26 — Verificador de Archivos de Solo Lectura

Objetivo
Detectar si un archivo tiene atributo de solo lectura.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        FileInfo fi = new FileInfo(ruta);
        lblResultado.Text = fi.IsReadOnly ? "Solo lectura" :
"Editable";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `FileInfo.IsReadOnly` indica si el archivo no puede modificarse.

Ejercicio 27 — Cambiador de Atributos (Solo Lectura)

Objetivo
Establecer o quitar el atributo de solo lectura.

Diseño

- `TextBox` `txtArchivo`
- `CheckBox` `chkSoloLectura`
- `Button` `btnAplicar`

Código

```
private void btnAplicar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        FileInfo fi = new FileInfo(ruta);
        fi.IsReadOnly = chkSoloLectura.Checked;
        MessageBox.Show("Atributo actualizado.", "Éxito");
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- `IsReadOnly` se ajusta según estado del `CheckBox`.

Ejercicio 28 — Buscador de Archivos Grandes (Tamaño mínimo)

Objetivo
Listar archivos mayores a un tamaño dado.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtTamañoMB`
- `Button` `btnBuscar`
- `ListBox` `lstArchivos`

Código

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta) &&
double.TryParse(txtTamañoMB.Text, out double tamañoMB))
    {
        lstArchivos.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta))
        {
            FileInfo fi = new FileInfo(file);
            if (fi.Length >= tamañoMB * 1024 * 1024)
                lstArchivos.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Ruta no válida o tamaño incorrecto.",
"Error");
    }
}
```

Explicación

- Convierte MB a bytes y compara tamaño de cada archivo.

Ejercicio 29 — Verificador de Directorio Vacío

Objetivo
Determinar si un directorio está vacío.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        bool vacio = Directory.GetFiles(ruta).Length == 0 &&
Directory.GetDirectories(ruta).Length == 0;
        lblResultado.Text = vacio ? "Directorio vacío" :
"Directorio no vacío";
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Comprueba existencia de archivos o subdirectorios para determinar vacío.

Ejercicio 30 — Creador de Archivos de Ejemplo

Objetivo
Generar varios archivos de ejemplo en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnCrear`
- `TextBox` `txtCantidad` (Número de archivos)

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta) && int.TryParse(txtCantidad.Text, out int cantidad))
    {
        for (int i = 1; i <= cantidad; i++)
        {
            string archivo = Path.Combine(ruta, $"Archivo{i}.txt");
            File.WriteAllText(archivo, $"Contenido del archivo {i}");
        }
        MessageBox.Show("Archivos de ejemplo creados.", "Éxito");
    }
    else
    {
        MessageBox.Show("Ruta no válida o cantidad incorrecta.", "Error");
    }
}
```

Explicación

- Crea múltiples archivos de texto con contenido numerado en el directorio especificado.

Ejercicio 31 — Comparador de Archivos (Tamaño y Contenido para pequeños)

Objetivo
Comparar dos archivos y determinar si son iguales.

Diseño

- TextBox `txtArchivo1`
- TextBox `txtArchivo2`
- Button `btnComparar`
- Label `lblResultado`

Código

```
private void btnComparar_Click(object sender, EventArgs e)
{
    string archivo1 = txtArchivo1.Text;
    string archivo2 = txtArchivo2.Text;

    if (File.Exists(archivo1) && File.Exists(archivo2))
    {
        FileInfo f1 = new FileInfo(archivo1);
        FileInfo f2 = new FileInfo(archivo2);

        if (f1.Length != f2.Length)
        {
            lblResultado.Text = "Archivos diferentes";
            return;
        }

        byte[] bytes1 = File.ReadAllBytes(archivo1);
        byte[] bytes2 = File.ReadAllBytes(archivo2);

        lblResultado.Text = bytes1.SequenceEqual(bytes2) ?
"Archivos idénticos" : "Archivos diferentes";
    }
    else
```

```
    {
        MessageBox.Show("Uno o ambos archivos no existen.",
"Error");
    }
}
```

Explicación

- Primero compara tamaño.
- Luego compara byte por byte usando `SequenceEqual`.

Ejercicio 32 — Listador de Archivos Modificados Hoy

Objetivo
Listar archivos modificados en la fecha actual.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnListar`
- `ListBox` `lstArchivos`

Código

```
private void btnListar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        lstArchivos.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta))
        {
            if (File.GetLastWriteTime(file).Date == 
DateTime.Today)
                lstArchivos.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Ruta no válida.", "Error");
    }
}
```

Explicación

- `File.GetLastWriteTime(file).Date` compara solo la fecha (sin hora).

Ejercicio 33 — Verificador de Ruta Válida (Path)

Objetivo
Determinar si una ruta de archivo es válida.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    try
    {
        string ruta = txtRuta.Text;
        string fullPath = Path.GetFullPath(ruta);
        lblResultado.Text = "Ruta válida";
    }
    catch
    {
        lblResultado.Text = "Ruta inválida";
    }
}
```

Explicación

- `Path.GetFullPath` lanza excepción si la ruta es inválida.

Ejercicio 34 — Extractor de Información de Ruta

Objetivo
Mostrar nombre de archivo, directorio y extensión.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnExtraer`
- `Label` `lblResultado`

Código

```
private void btnExtraer_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    lblResultado.Text = $"Directorio: 
{Path.GetDirectoryName(ruta)}\n" +
                        $"Archivo: {Path.GetFileName(ruta)}\n" +
                        $"Extensión: {Path.GetExtension(ruta)}";
}
```

Explicación

- `Path.GetDirectoryName`, `Path.GetFileName`, `Path.GetExtension` extraen partes de la ruta.

Ejercicio 35 — Creador de Archivo Temporal (Path.GetTempFileName)

Objetivo
Crear un archivo temporal y mostrar su ruta.

Diseño

- `Button` `btnCrear`
- `TextBox` `txtRutaTemp`

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string temp = Path.GetTempFileName();
    txtRutaTemp.Text = temp;
    MessageBox.Show("Archivo temporal creado.", "Éxito");
}
```

Explicación

- `Path.GetTempFileName` genera archivo único en carpeta temporal del sistema.

Ejercicio 36 — Verificador de Espacio de Archivo (KB, MB, GB)

Objetivo
Mostrar tamaño de un archivo en distintas unidades.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnVerificar`
- `Label` `lblTamaño`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (File.Exists(ruta))
    {
        long bytes = new FileInfo(ruta).Length;
        lblTamaño.Text = $"Bytes: {bytes}\nKB:
{bytes/1024.0:F2}\nMB: {bytes/1024.0/1024.0:F2}\nGB:
{bytes/1024.0/1024.0/1024.0:F2}";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- Convierte tamaño de bytes a KB, MB y GB usando divisiones sucesivas.

Ejercicio 37 — Buscador de Archivos Duplicados (por nombre)

Objetivo
Encontrar archivos con nombres repetidos en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnBuscar`
- `ListBox` `lstDuplicados`

Código

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        var nombres = new Dictionary<string, int>();
        foreach (var file in Directory.GetFiles(ruta))
        {
            string nombre = Path.GetFileName(file);
            if (nombres.ContainsKey(nombre)) nombres[nombre]++;
            else nombres[nombre] = 1;
        }

        lstDuplicados.Items.Clear();
        foreach (var kv in nombres.Where(kv => kv.Value > 1))
            lstDuplicados.Items.Add(kv.Key);
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Usa `Dictionary` para contar repeticiones de nombres de archivo.

Ejercicio 38 — Organizador por Extensión (Mover a carpetas)

Objetivo
Mover archivos a carpetas según su extensión.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnOrganizar`

Código

```
private void btnOrganizar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        foreach (var file in Directory.GetFiles(ruta))
        {
            string ext = Path.GetExtension(file).TrimStart('.');
            string carpeta = Path.Combine(ruta, ext);
            Directory.CreateDirectory(carpeta);
            string destino = Path.Combine(carpeta,
Path.GetFileName(file));
            File.Move(file, destino, true);
        }
        MessageBox.Show("Archivos organizados.", "Éxito");
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Crea carpetas por extensión y mueve archivos a ellas.

Ejercicio 39 — Verificador de Archivos del Sistema

Objetivo
Identificar archivos del sistema en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnVerificar`
- `ListBox` `lstArchivos`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        lstArchivos.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta))
        {
            FileInfo fi = new FileInfo(file);
            if (fi.Attributes.HasFlag(FileAttributes.System))
                lstArchivos.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- `FileAttributes.System` detecta archivos protegidos del sistema.

Ejercicio 40 — Creador de Estructura de Directorios Anidada

Objetivo
Crear carpetas anidadas de manera automática.

Diseño

- `TextBox` `txtRutaBase`
- `TextBox` `txtSubCarpetas` (Separadas por \)
- `Button` `btnCrear`

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string rutaBase = txtRutaBase.Text;
    string rutaCompleta = Path.Combine(rutaBase,
txtSubCarpetas.Text);
    Directory.CreateDirectory(rutaCompleta);
    MessageBox.Show("Estructura de directorios creada.", "Éxito");
}
```

Explicación

- `Directory.CreateDirectory` crea toda la estructura anidada automáticamente.

Ejercicio 41 — Limpiador de Archivos Temporales (.tmp)

Objetivo
Eliminar todos los archivos temporales `.tmp` de un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnLimpiar`

Código

```
private void btnLimpiar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        foreach (var file in Directory.GetFiles(ruta, "*.tmp"))
        {
            File.Delete(file);
        }
        MessageBox.Show("Archivos temporales eliminados.",
"Éxito");
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- `Directory.GetFiles(ruta, "*.tmp")` selecciona solo archivos con extensión `.tmp`.

Ejercicio 42 — Verificador de Acceso a Archivo (Escritura)

Objetivo
Comprobar si un archivo se puede escribir.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    try
    {
        using (FileStream fs = File.Open(ruta, FileMode.Open, FileAccess.Write)) { }
        lblResultado.Text = "Archivo editable";
    }
    catch
    {
        lblResultado.Text = "Archivo protegido o no accesible";
    }
}
```

Explicación

- Intenta abrir el archivo en modo escritura. Si falla, no es accesible.

Ejercicio 43 — Generador de Lista de Archivos (en un TXT)

Objetivo
Guardar la lista de archivos de un directorio en un archivo `.txt`.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtArchivoSalida`
- `Button` `btnGenerar`

Código

```
private void btnGenerar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string salida = txtArchivoSalida.Text;
    if (Directory.Exists(ruta))
    {
        var archivos = Directory.GetFiles(ruta);
        File.WriteAllLines(salida, archivos);
        MessageBox.Show("Lista de archivos generada.", "Éxito");
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- `File.WriteAllLines` escribe un array de strings en un archivo de texto.

Ejercicio 44 — Verificador de Archivos Recientes (Últimos 7 días)

Objetivo
Listar archivos modificados en la última semana.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnRecientes`
- `ListBox` `lstArchivos`

Código

```
private void btnRecientes_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        lstArchivos.Items.Clear();
        DateTime limite = DateTime.Today.AddDays(-7);
        foreach (var file in Directory.GetFiles(ruta))
        {
            if (File.GetLastWriteTime(file) >= limite)
                lstArchivos.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Compara fecha de última modificación con fecha límite (hace 7 días).

Ejercicio 45 — Creador de Backup Simple (docx, xlsx, pdf)

Objetivo
Copiar archivos de ciertos tipos a carpeta de backup.

Diseño

- `TextBox` `txtRutaOrigen`
- `TextBox` `txtRutaBackup`
- `Button` `btnBackup`

Código

```
private void btnBackup_Click(object sender, EventArgs e)
{
    string origen = txtRutaOrigen.Text;
    string backup = txtRutaBackup.Text;
    if (Directory.Exists(origen))
    {
        Directory.CreateDirectory(backup);
        string[] extensiones = { ".docx", ".xlsx", ".pdf" };
        foreach (var file in Directory.GetFiles(origen))
        {
            if (extensiones.Contains(Path.GetExtension(file)))
            {
                string destino = Path.Combine(backup,
Path.GetFileName(file));
                File.Copy(file, destino, true);
            }
        }
        MessageBox.Show("Backup completado.", "Éxito");
    }
    else
    {
        MessageBox.Show("Directorio origen no válido.", "Error");
    }
}
```

Explicación

- Filtra archivos por extensión y los copia a carpeta de backup.

Ejercicio 46 — Verificador de Archivos de Imagen

Objetivo
Listar archivos de imagen en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnListar`
- `ListBox` `lstImagenes`

Código

```
private void btnListar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string[] extensiones = { ".jpg", ".jpeg", ".png", ".gif", ".bmp" };
    if (Directory.Exists(ruta))
    {
        lstImagenes.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta))
        {
            if (extensiones.Contains(Path.GetExtension(file).ToLower()))
                lstImagenes.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Filtra archivos con extensiones de imágenes comunes.

Ejercicio 47 — Calculadora de Estadísticas de Directorio (Total archivos, tamaño, extensiones)

Objetivo
Mostrar cantidad de archivos, tamaño total y tipos de extensiones.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnEstadisticas`
- `TextBox` `txtResultado` (Multilínea)

Código

```
private void btnEstadisticas_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        var archivos = Directory.GetFiles(ruta);
        long tamaño = archivos.Sum(f => new FileInfo(f).Length);
        var extensiones = archivos.Select(f =>
Path.GetExtension(f)).Distinct();

        txtResultado.Text = $"Total archivos: {archivos.Length}\n"
+
                            $"Tamaño total: {tamaño / (1024 *
1024)} MB\n" +
                            $"Extensiones: {string.Join(", ",
extensiones)}";
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Suma tamaños y obtiene lista de extensiones únicas.

Ejercicio 48 — Renombrador Masivo (Añadir prefijo)

Objetivo
Agregar un prefijo a todos los archivos en un directorio.

Diseño

- `TextBox` `txtRuta`
- `TextBox` `txtPrefijo`
- `Button` `btnRenombrar`

Código

```
private void btnRenombrar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    string prefijo = txtPrefijo.Text;
    if (Directory.Exists(ruta))
    {
        foreach (var file in Directory.GetFiles(ruta))
        {
            string nombreNuevo = Path.Combine(ruta, prefijo +
Path.GetFileName(file));
            File.Move(file, nombreNuevo);
        }
        MessageBox.Show("Archivos renombrados.", "Éxito");
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- `File.Move` permite cambiar nombre sin alterar ubicación.

Ejercicio 49 — Verificador de Archivos de Log

Objetivo
Listar archivos `.log` en un directorio.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnListar`
- `ListBox` `lstLogs`

Código

```
private void btnListar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        lstLogs.Items.Clear();
        foreach (var file in Directory.GetFiles(ruta, "*.log"))
        {
            lstLogs.Items.Add(file);
        }
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- Busca únicamente archivos con extensión `.log`.

Ejercicio 50 — Monitor de Cambios en Directorio (FileSystemWatcher)

Objetivo
Detectar cambios (crear, modificar, eliminar) en un directorio en tiempo real.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnIniciar`
- `ListBox` `lstEventos`
- `FileSystemWatcher` `watcher`

Código

```
private void btnIniciar_Click(object sender, EventArgs e)
{
    string ruta = txtRuta.Text;
    if (Directory.Exists(ruta))
    {
        FileSystemWatcher watcher = new FileSystemWatcher(ruta);
        watcher.NotifyFilter = NotifyFilters.FileName |
NotifyFilters.LastWrite;
        watcher.Created += (s, ev) =>
lstEventos.Items.Add($"Creado: {ev.Name}");
        watcher.Changed += (s, ev) =>
lstEventos.Items.Add($"Modificado: {ev.Name}");
        watcher.Deleted += (s, ev) =>
lstEventos.Items.Add($"Eliminado: {ev.Name}");
        watcher.EnableRaisingEvents = true;
        MessageBox.Show("Monitoreo iniciado.", "Éxito");
    }
    else
    {
        MessageBox.Show("Directorio no válido.", "Error");
    }
}
```

Explicación

- `FileSystemWatcher` permite monitorear cambios en tiempo real en archivos de un directorio.

PARTE III — Archivos CSV y JSON

Ejercicio 1 — Lector Básico de CSV (DataGridView)

Objetivo
Cargar un archivo CSV en un `DataGridView`.

Diseño

- `Button` `btnCargar`
- `DataGridView` `dgvCSV`
- `OpenFileDialog` para seleccionar archivo

Código

```
private void btnCargar_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.Filter = "CSV Files|*.csv";
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        string[] lineas = File.ReadAllLines(ofd.FileName);
        dgvCSV.Rows.Clear();
        dgvCSV.Columns.Clear();

        if (lineas.Length > 0)
        {
            string[] encabezados = lineas[0].Split(',');
            foreach (string header in encabezados)
                dgvCSV.Columns.Add(header, header);

            for (int i = 1; i < lineas.Length; i++)
                dgvCSV.Rows.Add(lineas[i].Split(','));
        }
    }
}
```

Explicación

- Divide el CSV por comas.

- Primera fila se usa como encabezado, el resto como datos.

Ejercicio 2 — Generador de CSV Simple

Objetivo
Crear un archivo CSV desde un `DataGridView`.

Diseño

- `Button` `btnGuardar`
- `DataGridView` `dgvCSV`
- `SaveFileDialog` para seleccionar ubicación

Código

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "CSV Files|*.csv";
    if (sfd.ShowDialog() == DialogResult.OK)
    {
        List<string> lineas = new List<string>();
        string[] headers =
dgvCSV.Columns.Cast<DataGridViewColumn>().Select(c =>
c.HeaderText).ToArray();
        lineas.Add(string.Join(",", headers));

        foreach (DataGridViewRow row in dgvCSV.Rows)
        {
            if (!row.IsNewRow)
            {
                string[] datos =
row.Cells.Cast<DataGridViewCell>().Select(c => c.Value?.ToString()
?? "").ToArray();
                lineas.Add(string.Join(",", datos));
            }
        }

        File.WriteAllLines(sfd.FileName, lineas);
        MessageBox.Show("CSV generado.", "Éxito");
```

```
    }
}
```

Explicación

- Convierte filas y columnas en líneas CSV y guarda en archivo.

Ejercicio 3 — Contador de Filas en CSV (Excluyendo encabezado)

Objetivo
Contar el número de registros de un CSV (sin contar encabezado).

Diseño

- `Button` `btnContar`
- `TextBox` `txtRuta`
- `Label` `lblResultado`

Código

```
private void btnContar_Click(object sender, EventArgs e)
{
    if (File.Exists(txtRuta.Text))
    {
        int total = File.ReadAllLines(txtRuta.Text).Length - 1;
        lblResultado.Text = $"Filas: {total}";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- Restamos 1 para excluir la fila de encabezado.

Ejercicio 4 — Verificador de Estructura CSV (Columnas consistentes)

Objetivo
Comprobar que todas las filas tengan el mismo número de columnas que el encabezado.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    if (File.Exists(txtRuta.Text))
    {
        string[] lineas = File.ReadAllLines(txtRuta.Text);
        int columnas = lineas[0].Split(',').Length;
        bool consistente = lineas.Skip(1).All(l =>
l.Split(',').Length == columnas);
        lblResultado.Text = consistente ? "CSV consistente" : "CSV
inconsistente";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- Verifica que cada fila tenga mismo número de columnas que el encabezado.

Ejercicio 5 — Creador de JSON Simple

Objetivo
Generar un archivo JSON a partir de un objeto simple.

Diseño

- `Button` `btnCrearJSON`
- `TextBox` `txtRuta`

Código

```
using System.Text.Json;


private void btnCrearJSON_Click(object sender, EventArgs e)
{
    var persona = new { Nombre = "Ana", Edad = 25, Ciudad = "Zaragoza" };
    string json = JsonSerializer.Serialize(persona, new JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtRuta.Text, json);
    MessageBox.Show("JSON creado.", "Éxito");
}
```

Explicación

- `JsonSerializer.Serialize` convierte objeto a formato JSON legible.

Ejercicio 6 — Lector de JSON Simple

Objetivo
Leer un JSON y mostrar valores en `TextBox` o `Label`.

Diseño

- `Button` `btnLeerJSON`
- `TextBox` `txtRuta`
- `Label` `lblResultado`

Código

```
using System.Text.Json;


private void btnLeerJSON_Click(object sender, EventArgs e)
{
    if (File.Exists(txtRuta.Text))
    {
        string json = File.ReadAllText(txtRuta.Text);
        var persona =
JsonSerializer.Deserialize<Dictionary<string, object>>(json);
        lblResultado.Text = $"Nombre: {persona["Nombre"]}, Edad:
{persona["Edad"]}, Ciudad: {persona["Ciudad"]}";
    }
    else
    {
        MessageBox.Show("Archivo no encontrado.", "Error");
    }
}
```

Explicación

- Deserializa JSON en un `Dictionary` para acceder a campos fácilmente.

Ejercicio 7 — Convertidor CSV a JSON

Objetivo
Convertir un archivo CSV a JSON.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtJSON`
- `Button` `btnConvertir`

Código

```
using System.Text.Json;

private void btnConvertir_Click(object sender, EventArgs e)
{
    if (File.Exists(txtCSV.Text))
    {
        string[] lineas = File.ReadAllLines(txtCSV.Text);
        string[] headers = lineas[0].Split(',');
        var lista = new List<Dictionary<string, string>>();

        for (int i = 1; i < lineas.Length; i++)
        {
            string[] valores = lineas[i].Split(',');
            var dict = new Dictionary<string, string>();
            for (int j = 0; j < headers.Length; j++)
                dict[headers[j]] = valores[j];
            lista.Add(dict);
        }

        string json = JsonSerializer.Serialize(lista, new
JsonSerializerOptions { WriteIndented = true });
        File.WriteAllText(txtJSON.Text, json);
        MessageBox.Show("CSV convertido a JSON.", "Éxito");
    }
    else
```

```
    {
        MessageBox.Show("CSV no encontrado.", "Error");
    }
}
```

Explicación

- Cada fila del CSV se convierte en un diccionario y luego se serializa a JSON.

Ejercicio 8 — Buscador de Texto en CSV

Objetivo
Buscar una palabra en todas las celdas del CSV.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtBuscar`
- `Button` `btnBuscar`
- `ListBox` `lstResultados`

Código

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    if (File.Exists(txtCSV.Text))
    {
        string buscar = txtBuscar.Text;
        lstResultados.Items.Clear();
        string[] lineas = File.ReadAllLines(txtCSV.Text);
        for (int i = 0; i < lineas.Length; i++)
        {
            if (lineas[i].Contains(buscar))
                lstResultados.Items.Add($"Fila {i+1}:
{lineas[i]}");
        }
    }
    else
    {
        MessageBox.Show("CSV no encontrado.", "Error");
    }
}
```

Explicación

- Recorre cada fila y comprueba si contiene el texto buscado.

Ejercicio 9 — Validador de JSON (JsonDocument.Parse)

Objetivo
Comprobar si un archivo JSON es válido.

Diseño

- `TextBox` `txtRuta`
- `Button` `btnValidar`
- `Label` `lblResultado`

Código

```
using System.Text.Json;

private void btnValidar_Click(object sender, EventArgs e)
{
    try
    {
        string json = File.ReadAllText(txtRuta.Text);
        JsonDocument.Parse(json);
        lblResultado.Text = "JSON válido";
    }
    catch
    {
        lblResultado.Text = "JSON inválido";
    }
}
```

Explicación

- `JsonDocument.Parse` lanza excepción si JSON no tiene formato correcto.

Ejercicio 10 — Editor de CSV (Cargar y Guardar desde DataGridView)

Objetivo
Permitir editar un CSV en `DataGridView` y guardar cambios.

Diseño

- `Button` `btnCargar`
- `Button` `btnGuardar`
- `DataGridView` `dgvCSV`
- `TextBox` `txtCSV`

Código

```
private void btnCargar_Click(object sender, EventArgs e)
{
    if (File.Exists(txtCSV.Text))
    {
        string[] lineas = File.ReadAllLines(txtCSV.Text);
        dgvCSV.Rows.Clear();
        dgvCSV.Columns.Clear();

        string[] headers = lineas[0].Split(',');
        foreach (string h in headers) dgvCSV.Columns.Add(h, h);

        for (int i = 1; i < lineas.Length; i++)
            dgvCSV.Rows.Add(lineas[i].Split(','));
    }
}

private void btnGuardar_Click(object sender, EventArgs e)
{
    List<string> lineas = new List<string>();
    string[] headers =
dgvCSV.Columns.Cast<DataGridViewColumn>().Select(c =>
c.HeaderText).ToArray();
    lineas.Add(string.Join(",", headers));
```

```
    foreach (DataGridViewRow row in dgvCSV.Rows)
    {
        if (!row.IsNewRow)
        {
            string[] datos =
row.Cells.Cast<DataGridViewCell>().Select(c => c.Value?.ToString()
?? "").ToArray();
            lineas.Add(string.Join(",", datos));
        }
    }

    File.WriteAllLines(txtCSV.Text, lineas);
    MessageBox.Show("CSV guardado.", "Éxito");
}
```

Explicación

- Permite cargar un CSV para editarlo en `DataGridView` y luego sobrescribir el archivo.

Ejercicio 11 — Generador de Datos de Prueba CSV (Aleatorios)

Objetivo
Crear un CSV con datos aleatorios para pruebas.

Diseño

- `TextBox` `txtCSV`
- `Button` `btnGenerar`
- `TextBox` `txtFilas`

Código

```
private void btnGenerar_Click(object sender, EventArgs e)
{
    int filas = int.Parse(txtFilas.Text);
    List<string> lineas = new List<string>();
    lineas.Add("ID,Nombre,Edad"); // encabezado

    Random rnd = new Random();
    for (int i = 1; i <= filas; i++)
    {
        string nombre = "Persona" + i;
        int edad = rnd.Next(18, 60);
        lineas.Add($"{i},{nombre},{edad}");
    }

    File.WriteAllLines(txtCSV.Text, lineas);
    MessageBox.Show("CSV generado con datos aleatorios.",
"Éxito");
}
```

Explicación

- Genera IDs secuenciales, nombres y edades aleatorias.

Ejercicio 12 — Extractor de Columnas CSV

Objetivo
Extraer una columna específica de un CSV y mostrarla en un `ListBox`.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtColumna`
- `Button` `btnExtraer`
- `ListBox` `lstColumna`

Código

```
private void btnExtraer_Click(object sender, EventArgs e)
{
    string[] lineas = File.ReadAllLines(txtCSV.Text);
    string[] headers = lineas[0].Split(',');
    int indice = Array.IndexOf(headers, txtColumna.Text);

    if (indice < 0)
    {
        MessageBox.Show("Columna no encontrada", "Error");
        return;
    }

    lstColumna.Items.Clear();
    for (int i = 1; i < lineas.Length; i++)
    {
        string[] datos = lineas[i].Split(',');
        lstColumna.Items.Add(datos[indice]);
    }
}
```

Explicación

- Localiza el índice de la columna por nombre y extrae sus valores.

Ejercicio 13 — Analizador de JSON con Clases (Deserialize)

Objetivo
Leer JSON en un objeto fuertemente tipado.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnLeer`
- `Label` `lblResultado`

Código

```
using System.Text.Json;

public class Persona
{
    public string Nombre { get; set; }
    public int Edad { get; set; }
    public string Ciudad { get; set; }
}

private void btnLeer_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    Persona p = JsonSerializer.Deserialize<Persona>(json);
    lblResultado.Text = $"Nombre: {p.Nombre}, Edad: {p.Edad},
Ciudad: {p.Ciudad}";
}
```

Explicación

- Permite acceder a los campos del JSON mediante propiedades de clase.

Ejercicio 14 — Filtrador de CSV (Basado en columna/valor)

Objetivo
Mostrar filas que cumplen un criterio en una columna específica.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtColumna`
- `TextBox` `txtValor`
- `Button` `btnFiltrar`
- `ListBox` `lstFiltrado`

Código

```
private void btnFiltrar_Click(object sender, EventArgs e)
{
    string[] lineas = File.ReadAllLines(txtCSV.Text);
    string[] headers = lineas[0].Split(',');
    int indice = Array.IndexOf(headers, txtColumna.Text);

    if (indice < 0) { MessageBox.Show("Columna no encontrada");
return; }

    lstFiltrado.Items.Clear();
    for (int i = 1; i < lineas.Length; i++)
    {
        string[] datos = lineas[i].Split(',');
        if (datos[indice] == txtValor.Text)
            lstFiltrado.Items.Add(lineas[i]);
    }
}
```

Explicación

- Permite filtrar filas según un valor exacto en la columna especificada.

Ejercicio 15 — Creador de JSON Array (Lista de objetos)

Objetivo
Generar un JSON que contenga una lista de objetos.

Diseño

- `Button` `btnCrearJSON`
- `TextBox` `txtJSON`

Código

```
using System.Text.Json;


private void btnCrearJSON_Click(object sender, EventArgs e)
{
    var lista = new List<Persona>
    {
        new Persona { Nombre="Ana", Edad=25, Ciudad="Zaragoza" },
        new Persona { Nombre="Luis", Edad=30, Ciudad="Madrid" },
        new Persona { Nombre="Marta", Edad=28,
Ciudad="Barcelona" }
    };


    string json = JsonSerializer.Serialize(lista, new
JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSON.Text, json);
    MessageBox.Show("JSON Array creado.", "Éxito");
}
```

Explicación

- Cada objeto de la lista se convierte en un elemento del array JSON.

Ejercicio 16 — Calculadora de Estadísticas CSV (Suma, promedio, máx.)

Objetivo
Calcular suma, promedio y valor máximo de una columna numérica.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtColumna`
- `Button` `btnCalcular`
- `Label` `lblResultado`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    string[] lineas = File.ReadAllLines(txtCSV.Text);
    string[] headers = lineas[0].Split(',');
    int indice = Array.IndexOf(headers, txtColumna.Text);
    if (indice < 0) { MessageBox.Show("Columna no encontrada");
return; }

    List<double> numeros = new List<double>();
    for (int i = 1; i < lineas.Length; i++)
        numeros.Add(double.Parse(lineas[i].Split(',')[indice]));

    double suma = numeros.Sum();
    double promedio = numeros.Average();
    double max = numeros.Max();

    lblResultado.Text = $"Suma: {suma}, Promedio: {promedio}, Máx:
{max}";
}
```

Explicación

- Convierte los valores de la columna a números y calcula estadísticas.

Ejercicio 17 — Combinador de Archivos CSV

Objetivo
Unir varios archivos CSV en uno solo.

Diseño

- `TextBox` `txtCSV1`
- `TextBox` `txtCSV2`
- `TextBox` `txtSalida`
- `Button` `btnCombinar`

Código

```
private void btnCombinar_Click(object sender, EventArgs e)
{
    var lineas1 = File.ReadAllLines(txtCSV1.Text);
    var lineas2 = File.ReadAllLines(txtCSV2.Text);

    List<string> combinadas = new List<string>();
    combinadas.Add(lineas1[0]); // encabezado
    combinadas.AddRange(lineas1.Skip(1));
    combinadas.AddRange(lineas2.Skip(1)); // evitar duplicar encabezado

    File.WriteAllLines(txtSalida.Text, combinadas);
    MessageBox.Show("CSV combinados.", "Éxito");
}
```

Explicación

- Mantiene solo un encabezado y concatena las filas de ambos CSV.

Ejercicio 18 — Editor de JSON Interactivo (Reemplazo simple)

Objetivo
Modificar un valor en un JSON y guardar los cambios.

Diseño

- `TextBox` `txtJSON`
- `TextBox` `txtClave`
- `TextBox` `txtValor`
- `Button` `btnModificar`

Código

```
using System.Text.Json;


private void btnModificar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    var dict = JsonSerializer.Deserialize<Dictionary<string, object>>(json);
    dict[txtClave.Text] = txtValor.Text;
    string nuevoJson = JsonSerializer.Serialize(dict, new JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSON.Text, nuevoJson);
    MessageBox.Show("JSON modificado.", "Éxito");
}
```

Explicación

- Deserializa a `Dictionary`, cambia el valor y vuelve a serializar.

Ejercicio 19 — Generador de CSV desde Formulario

Objetivo
Crear un CSV tomando datos de controles `TextBox`.

Diseño

- `TextBox` `txtNombre`, `txtEdad`, `txtCiudad`
- `TextBox` `txtCSV`
- `Button` `btnAgregar`

Código

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    bool existe = File.Exists(txtCSV.Text);
    using (StreamWriter sw = new StreamWriter(txtCSV.Text, true))
    {
        if (!existe)
            sw.WriteLine("Nombre,Edad,Ciudad");
        sw.WriteLine($"{txtNombre.Text},{txtEdad.Text},
{txtCiudad.Text}");
    }
    MessageBox.Show("Fila agregada al CSV.", "Éxito");
}
```

Explicación

- Si el CSV no existe, crea encabezado; luego agrega la nueva fila.

Ejercicio 20 — Visualizador de JSON con TreeView (Estructura)

Objetivo
Mostrar la estructura de un JSON en un `TreeView`.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnCargar`
- `TreeView` `treeJSON`

Código

```
using System.Text.Json;

private void btnCargar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    treeJSON.Nodes.Clear();

    void AgregarNodo(JsonElement element, TreeNode node)
    {
        switch (element.ValueKind)
        {
            case JsonValueKind.Object:
                foreach (var prop in element.EnumerateObject())
                {
                    TreeNode child = node.Nodes.Add(prop.Name);
                    AgregarNodo(prop.Value, child);
                }
                break;
            case JsonValueKind.Array:
                int index = 0;
                foreach (var item in element.EnumerateArray())
                {
                    TreeNode child = node.Nodes.Add($"[{index}]");
                    AgregarNodo(item, child);
```

```
                    index++;
                }
                break;
            default:
                node.Nodes.Add(element.ToString());
                break;
        }
    }

    TreeNode root = treeJSON.Nodes.Add("JSON");
    AgregarNodo(JsonDocument.Parse(json).RootElement, root);
    treeJSON.ExpandAll();
}
```

Explicación

- Recursión para recorrer objetos y arrays, mostrando jerarquía en `TreeView`.

Ejercicio 21 — Convertidor de JSON a CSV

Objetivo
Convertir un archivo JSON (array de objetos) a CSV.

Diseño

- `TextBox` `txtJSON`
- `TextBox` `txtCSV`
- `Button` `btnConvertir`

Código

```
using System.Text.Json;


private void btnConvertir_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    var lista = JsonSerializer.Deserialize<List<Dictionary<string, object>>>(json);


    if (lista.Count == 0) return;


    var headers = lista[0].Keys.ToArray();
    List<string> lineas = new List<string> { string.Join(",", headers) };


    foreach (var item in lista)
    {
        lineas.Add(string.Join(",", headers.Select(h => item[h].ToString())));
    }


    File.WriteAllLines(txtCSV.Text, lineas);
    MessageBox.Show("JSON convertido a CSV.", "Éxito");
}
```

Explicación

- Toma el primer objeto para los encabezados y recorre la lista generando las filas.

Ejercicio 22 — Validador de CSV con Encabezados Específicos

Objetivo
Comprobar que un CSV contenga todos los encabezados requeridos.

Diseño

- `TextBox` `txtCSV`
- `Button` `btnValidar`
- `Label` `lblResultado`

Código

```
private void btnValidar_Click(object sender, EventArgs e)
{
    string[] requeridos = { "Nombre", "Edad", "Ciudad" };
    string[] headers = File.ReadAllLines(txtCSV.Text)
[0].Split(',');

    bool ok = requeridos.All(r => headers.Contains(r));
    lblResultado.Text = ok ? "CSV válido" : "CSV inválido";
}
```

Explicación

- Verifica que todos los encabezados obligatorios estén presentes.

Ejercicio 23 — Generador de Configuración JSON

Objetivo
Crear un JSON de configuración con claves y valores.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnCrear`

Código

```
using System.Text.Json;


private void btnCrear_Click(object sender, EventArgs e)
{
    var config = new Dictionary<string, object>
    {
        { "Tema", "Oscuro" },
        { "Idioma", "ES" },
        { "MostrarNotificaciones", true },
        { "Volumen", 80 }
    };


    string json = JsonSerializer.Serialize(config, new
JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSON.Text, json);
    MessageBox.Show("Configuración JSON creada.", "Éxito");
}
```

Explicación

- Permite crear fácilmente archivos de configuración para aplicaciones.

Ejercicio 24 — Lector de Configuración JSON (Deserialización a clase)

Objetivo
Leer un archivo de configuración JSON en una clase.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnLeer`
- `Label` `lblResultado`

Código

```
using System.Text.Json;

public class Config
{
    public string Tema { get; set; }
    public string Idioma { get; set; }
    public bool MostrarNotificaciones { get; set; }
    public int Volumen { get; set; }
}

private void btnLeer_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    Config cfg = JsonSerializer.Deserialize<Config>(json);
    lblResultado.Text = $"Tema: {cfg.Tema}, Idioma: {cfg.Idioma},
Notif: {cfg.MostrarNotificaciones}, Vol: {cfg.Volumen}";
}
```

Explicación

- Acceso seguro a la configuración mediante propiedades de clase.

Ejercicio 25 — Separador de CSV (Dividir en partes por número de filas)

Objetivo
Dividir un CSV grande en varios archivos pequeños.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtFilas`
- `Button` `btnSeparar`

Código

```
private void btnSeparar_Click(object sender, EventArgs e)
{
    string[] lineas = File.ReadAllLines(txtCSV.Text);
    int filasPorArchivo = int.Parse(txtFilas.Text);
    string header = lineas[0];

    int count = 0, fileIndex = 1;
    List<string> buffer = new List<string> { header };

    for (int i = 1; i < lineas.Length; i++)
    {
        buffer.Add(lineas[i]);
        count++;
        if (count == filasPorArchivo || i == lineas.Length - 1)
        {
            File.WriteAllLines($"Parte_{fileIndex}.csv", buffer);
            buffer = new List<string> { header };
            count = 0;
            fileIndex++;
        }
    }
    MessageBox.Show("CSV separado en partes.", "Éxito");
}
```

Explicación

- Crea archivos con el número de filas indicado manteniendo el encabezado.

Ejercicio 26 — Buscador en JSON (Recursivo en elementos)

Objetivo
Buscar un valor dentro de un JSON anidado.

Diseño

- `TextBox` `txtJSON`
- `TextBox` `txtBuscar`
- `Button` `btnBuscar`
- `ListBox` `lstResultados`

Código

```
using System.Text.Json;

private void btnBuscar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    JsonDocument doc = JsonDocument.Parse(json);
    lstResultados.Items.Clear();

    void Buscar(JsonElement element, string path)
    {
        switch (element.ValueKind)
        {
            case JsonValueKind.Object:
                foreach (var prop in element.EnumerateObject())
                    Buscar(prop.Value, path + "." + prop.Name);
                break;
            case JsonValueKind.Array:
                int index = 0;
                foreach (var item in element.EnumerateArray())
                {
                    Buscar(item, path + $"[{index}]");
                    index++;
                }
```

```
                break;
            default:
                if (element.ToString().Contains(txtBuscar.Text))
                    lstResultados.Items.Add($"{path}: {element}");
                break;
        }
    }

    Buscar(doc.RootElement, "Root");
}
```

Explicación

- Recursión para encontrar valores en cualquier nivel del JSON.

Ejercicio 27 — Generador de JSON desde CSV (Con Dictionary)

Objetivo
Crear un JSON a partir de un CSV usando diccionarios.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtJSON`
- `Button` `btnConvertir`

Código

```
using System.Text.Json;


private void btnConvertir_Click(object sender, EventArgs e)
{
    string[] lineas = File.ReadAllLines(txtCSV.Text);
    string[] headers = lineas[0].Split(',');


    List<Dictionary<string, string>> lista = new
List<Dictionary<string, string>>();
    for (int i = 1; i < lineas.Length; i++)
    {
        var dict = new Dictionary<string, string>();
        string[] valores = lineas[i].Split(',');
        for (int j = 0; j < headers.Length; j++)
            dict[headers[j]] = valores[j];
        lista.Add(dict);
    }


    string json = JsonSerializer.Serialize(lista, new
JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSON.Text, json);
    MessageBox.Show("CSV convertido a JSON.", "Éxito");
}
```

Explicación

- Cada fila se convierte en un diccionario y luego se serializa a JSON.

Ejercicio 28 — Calculadora de Tamaño de Archivos (Con datos CSV/JSON)

Objetivo
Sumar tamaños de archivos listados en CSV o JSON.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnCalcular`
- `Label` `lblResultado`

Código

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    double total = 0;
    if (txtArchivo.Text.EndsWith(".csv"))
    {
        var lines = File.ReadAllLines(txtArchivo.Text).Skip(1);
        foreach (var l in lines)
        {
            double tamaño = double.Parse(l.Split(',')[1]);
            total += tamaño;
        }
    }
    else if (txtArchivo.Text.EndsWith(".json"))
    {
        var json = File.ReadAllText(txtArchivo.Text);
        var lista =
JsonSerializer.Deserialize<List<Dictionary<string,
object>>>(json);
        foreach (var dict in lista)
            total += double.Parse(dict["Tamaño"].ToString());
    }

    lblResultado.Text = $"Tamaño total: {total} KB";
}
```

Explicación

- Funciona con CSV o JSON que contengan columna/propiedad “Tamaño”.

Ejercicio 29 — Comparador de Archivos CSV (Diferencias línea por línea)

Objetivo
Comparar dos CSV y mostrar filas diferentes.

Diseño

- `TextBox` `txtCSV1`
- `TextBox` `txtCSV2`
- `Button` `btnComparar`
- `ListBox` `lstDiferencias`

Código

```
private void btnComparar_Click(object sender, EventArgs e)
{
    string[] file1 = File.ReadAllLines(txtCSV1.Text);
    string[] file2 = File.ReadAllLines(txtCSV2.Text);
    lstDiferencias.Items.Clear();

    int max = Math.Max(file1.Length, file2.Length);
    for (int i = 0; i < max; i++)
    {
        string f1 = i < file1.Length ? file1[i] : "";
        string f2 = i < file2.Length ? file2[i] : "";
        if (f1 != f2)
            lstDiferencias.Items.Add($"Fila {i + 1} dif: '{f1}' vs
'{f2}'");
    }
}
```

Explicación

- Compara línea por línea y muestra diferencias con índice de fila.

Ejercicio 30 — Generador de JSON con Datos Aleatorios para Pruebas

Objetivo
Crear JSON con datos aleatorios para pruebas de forma automática.

Diseño

- TextBox `txtJSON`
- TextBox `txtRegistros`
- Button `btnGenerar`

Código

```
using System.Text.Json;

private void btnGenerar_Click(object sender, EventArgs e)
{
    int n = int.Parse(txtRegistros.Text);
    Random rnd = new Random();
    List<Persona> lista = new List<Persona>();

    for (int i = 0; i < n; i++)
    {
        lista.Add(new Persona
        {
            Nombre = "Persona" + i,
            Edad = rnd.Next(18, 60),
            Ciudad = "Ciudad" + rnd.Next(1, 10)
        });
    }

    string json = JsonSerializer.Serialize(lista, new
JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSON.Text, json);
    MessageBox.Show("JSON generado con datos aleatorios.",
"Éxito");
}
```

Explicación

- Permite generar fácilmente grandes volúmenes de datos para pruebas de JSON.

Ejercicio 31 — Limpiador de CSV (Normalizar espacios y eliminar vacíos)

Objetivo
Eliminar espacios innecesarios y filas vacías en un CSV.

Diseño

- `TextBox` `txtCSV`
- `Button` `btnLimpiar`

Código

```
private void btnLimpiar_Click(object sender, EventArgs e)
{
    var lineas = File.ReadAllLines(txtCSV.Text)
        .Select(l => l.Trim())
        .Where(l => !string.IsNullOrEmpty(l))
        .ToArray();
    File.WriteAllLines(txtCSV.Text, lineas);
    MessageBox.Show("CSV limpiado.", "Éxito");
}
```

Explicación

- Se eliminan espacios al inicio y fin, y se descartan líneas vacías.

Ejercicio 32 — Contador de Elementos JSON (Recursivo)

Objetivo
Contar todos los elementos en un JSON, incluyendo objetos y arrays anidados.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnContar`
- `Label` `lblResultado`

Código

```
using System.Text.Json;

private void btnContar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    JsonDocument doc = JsonDocument.Parse(json);
    int total = 0;

    void Contar(JsonElement element)
    {
        total++;
        switch (element.ValueKind)
        {
            case JsonValueKind.Object:
                foreach (var prop in element.EnumerateObject())
Contar(prop.Value);
                break;
            case JsonValueKind.Array:
                foreach (var item in element.EnumerateArray())
Contar(item);
                break;
        }
    }

    Contar(doc.RootElement);
```

```
    lblResultado.Text = $"Total elementos: {total}";
}
```

Explicación

- La recursión permite contar todos los elementos en cualquier nivel del JSON.

Ejercicio 33 — Exportador de DataGridView a CSV

Objetivo
Guardar el contenido de un `DataGridView` en un archivo CSV.

Diseño

- `Button` `btnExportar`
- `DataGridView` `dgvCSV`
- `TextBox` `txtArchivo`

Código

```
private void btnExportar_Click(object sender, EventArgs e)
{
    List<string> lineas = new List<string>();
    string[] headers =
dgvCSV.Columns.Cast<DataGridViewColumn>().Select(c =>
c.HeaderText).ToArray();
    lineas.Add(string.Join(",", headers));

    foreach (DataGridViewRow row in dgvCSV.Rows)
    {
        if (!row.IsNewRow)
        {
            string[] valores =
row.Cells.Cast<DataGridViewCell>().Select(c => c.Value?.ToString()
?? "").ToArray();
            lineas.Add(string.Join(",", valores));
        }
    }

    File.WriteAllLines(txtArchivo.Text, lineas);
    MessageBox.Show("DataGridView exportado a CSV.", "Éxito");
}
```

Explicación

- Convierte cada fila y columna en una línea CSV.

Ejercicio 34 — Creador de JSON Anidado (Estructura compleja)

Objetivo
Generar un JSON con objetos anidados y arrays.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnCrear`

Código

```
using System.Text.Json;


private void btnCrear_Click(object sender, EventArgs e)
{
    var data = new
    {
        Empresa = "TechCorp",
        Empleados = new[]
        {
            new { Nombre="Ana", Edad=25, Roles=new[]{ "Dev",
"QA" } },
            new { Nombre="Luis", Edad=30, Roles=new[]{ "Manager" }
}
        }
    };


    string json = JsonSerializer.Serialize(data, new
JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSON.Text, json);
    MessageBox.Show("JSON anidado creado.", "Éxito");
}
```

Explicación

- Permite crear estructuras complejas con objetos dentro de arrays.

Ejercicio 35 — Verificador de Codificación CSV (UTF-8 vs ASCII)

Objetivo
Detectar si un CSV está en UTF-8 o ASCII.

Diseño

- `TextBox` `txtCSV`
- `Button` `btnVerificar`
- `Label` `lblResultado`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    byte[] bytes = File.ReadAllBytes(txtCSV.Text);
    try
    {
        string contenido =
System.Text.Encoding.UTF8.GetString(bytes);
        lblResultado.Text = "Codificación: UTF-8";
    }
    catch
    {
        lblResultado.Text = "Codificación: ASCII u otra";
    }
}
```

Explicación

- Intenta decodificar en UTF-8; si falla, se asume otra codificación.

Ejercicio 36 — Modificador de JSON (Búsqueda de camino simple)

Objetivo
Modificar un valor en un JSON dado un “camino” simple como `Empleado[0].Nombre`.

Diseño

- `TextBox` `txtJSON`
- `TextBox` `txtCamino`
- `TextBox` `txtValor`
- `Button` `btnModificar`

Código (simplificado para objetos planos)

```
using System.Text.Json;


private void btnModificar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    var dict = JsonSerializer.Deserialize<Dictionary<string, object>>(json);
    dict[txtCamino.Text] = txtValor.Text; // solo para nivel 1
    File.WriteAllText(txtJSON.Text, JsonSerializer.Serialize(dict, new JsonSerializerOptions { WriteIndented = true }));
    MessageBox.Show("JSON modificado.", "Éxito");
}
```

Explicación

- Modifica un valor dado un nombre de propiedad (nivel 1).

Ejercicio 37 — Generador de CSV con Encabezados Personalizados

Objetivo
Crear un CSV con encabezados definidos por el usuario.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtEncabezados` (separados por comas)
- `Button` `btnCrear`

Código

```
private void btnCrear_Click(object sender, EventArgs e)
{
    string[] headers = txtEncabezados.Text.Split(',');
    File.WriteAllLines(txtCSV.Text, new[] { string.Join(",", headers) });
    MessageBox.Show("CSV con encabezados personalizados creado.", "Éxito");
}
```

Explicación

- Permite definir encabezados antes de agregar datos.

Ejercicio 38 — Validador de Tipos en JSON (Mostrar tipos de datos)

Objetivo
Mostrar el tipo de cada valor en un JSON.

Diseño

- TextBox `txtJSON`
- Button `btnAnalizar`
- ListBox `lstTipos`

Código

```
using System.Text.Json;

private void btnAnalizar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    JsonDocument doc = JsonDocument.Parse(json);
    lstTipos.Items.Clear();

    void Analizar(JsonElement element, string path)
    {
        switch (element.ValueKind)
        {
            case JsonValueKind.Object:
                foreach (var prop in element.EnumerateObject())
Analizar(prop.Value, path + "." + prop.Name);
                break;
            case JsonValueKind.Array:
                int i = 0;
                foreach (var item in element.EnumerateArray())
                {
                    Analizar(item, path + $"[{i}]");
                    i++;
                }
                break;
            default:
```

```
                lstTipos.Items.Add($"{path}:
{element.ValueKind}");
                break;
        }
    }

    Analizar(doc.RootElement, "Root");
}
```

Explicación

- Recursivamente identifica tipos (`String`, `Number`, `Boolean`, etc.) de todos los valores.

Ejercicio 39 — Convertidor de Encoding CSV

Objetivo
Cambiar la codificación de un CSV (por ejemplo de UTF-8 a ASCII).

Diseño

- `TextBox` `txtCSV`
- `Button` `btnConvertir`

Código

```
private void btnConvertir_Click(object sender, EventArgs e)
{
    string contenido = File.ReadAllText(txtCSV.Text,
System.Text.Encoding.UTF8);
    File.WriteAllText(txtCSV.Text, contenido,
System.Text.Encoding.ASCII);
    MessageBox.Show("CSV convertido a ASCII.", "Éxito");
}
```

Explicación

- Lee el archivo en UTF-8 y lo reescribe en ASCII.

Ejercicio 40 — Analizador de Estructura CSV (Mostrar tipos inferidos)

Objetivo
Detectar el tipo de cada columna en un CSV (String, Número, Fecha).

Diseño

- `TextBox` `txtCSV`
- `Button` `btnAnalizar`
- `ListBox` `lstTipos`

Código

```
private void btnAnalizar_Click(object sender, EventArgs e)
{
    var lines = File.ReadAllLines(txtCSV.Text);
    var headers = lines[0].Split(',');
    var tipos = new string[headers.Length];

    for (int i = 0; i < headers.Length; i++)
```

```
    {
        bool esNum = true, esFecha = true;
        for (int j = 1; j < lines.Length; j++)
        {
            string val = lines[j].Split(',')[i];
            if (!double.TryParse(val, out _)) esNum = false;
            if (!DateTime.TryParse(val, out _)) esFecha = false;
        }
        tipos[i] = esNum ? "Número" : esFecha ? "Fecha" :
"String";
    }

    lstTipos.Items.Clear();
    for (int i = 0; i < headers.Length; i++)
        lstTipos.Items.Add($"{headers[i]}: {tipos[i
]}");
}
```

Explicación

- Recorre cada columna y prueba cada valor para inferir su tipo.

Ejercicio 41 — Generador de JSON Schema (Básico)

Objetivo
Crear un JSON Schema sencillo para validar la estructura de un JSON.

Diseño

- `TextBox` `txtJSONSchema`
- `Button` `btnCrearSchema`

Código

```
using System.Text.Json;


private void btnCrearSchema_Click(object sender, EventArgs e)
{
    var schema = new
    {
        type = "object",
        properties = new
        {
            Nombre = new { type = "string" },
            Edad = new { type = "integer" },
            Ciudad = new { type = "string" }
        },
        required = new[] { "Nombre", "Edad" }
    };


    string json = JsonSerializer.Serialize(schema, new
JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(txtJSONSchema.Text, json);
    MessageBox.Show("JSON Schema creado.", "Éxito");
}
```

Explicación

- Genera un esquema básico indicando tipos de propiedades y campos obligatorios.

Ejercicio 42 — Filtrador de JSON (Búsqueda de propiedad/valor)

Objetivo
Filtrar objetos en un JSON array por un valor específico en una propiedad.

Diseño

- TextBox `txtJSON`
- TextBox `txtPropiedad`
- TextBox `txtValor`
- Button `btnFiltrar`
- ListBox `lstFiltrado`

Código

```
using System.Text.Json;


private void btnFiltrar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    var lista = JsonSerializer.Deserialize<List<Dictionary<string, object>>>(json);
    lstFiltrado.Items.Clear();

    foreach (var obj in lista)
    {
        if (obj.ContainsKey(txtPropiedad.Text) && obj[txtPropiedad.Text].ToString() == txtValor.Text)
        {
            lstFiltrado.Items.Add(JsonSerializer.Serialize(obj));
        }
    }
}
```

Explicación

- Permite filtrar elementos de un JSON array según un valor exacto de propiedad.

Ejercicio 43 — Calculadora de Hash de Archivos (MD5)

Objetivo
Calcular el hash MD5 de un archivo para verificar integridad.

Diseño

- `TextBox` `txtArchivo`
- `Button` `btnHash`
- `Label` `lblHash`

Código

```
using System.Security.Cryptography;
using System.Text;

private void btnHash_Click(object sender, EventArgs e)
{
    using (var md5 = MD5.Create())
    {
        using (var stream = File.OpenRead(txtArchivo.Text))
        {
            var hash = md5.ComputeHash(stream);
            lblHash.Text =
BitConverter.ToString(hash).Replace("-", "").ToLower();
        }
    }
}
```

Explicación

- Permite generar un hash MD5 para comparar versiones o verificar integridad.

Ejercicio 44 — Generador de Reporte CSV (Estadísticas por columna)

Objetivo
Crear un CSV resumen con estadísticas (suma, promedio, máximo) por columna numérica.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtReporte`
- `Button` `btnGenerarReporte`

Código

```
private void btnGenerarReporte_Click(object sender, EventArgs e)
{
    var lines = File.ReadAllLines(txtCSV.Text);
    var headers = lines[0].Split(',');
    List<string> reporte = new List<string>
{ "Columna,Suma,Promedio,Max" };

    for (int i = 0; i < headers.Length; i++)
    {
        List<double> valores = new List<double>();
        for (int j = 1; j < lines.Length; j++)
        {
            if (double.TryParse(lines[j].Split(',')[i], out double
val))
                valores.Add(val);
        }
        if (valores.Count > 0)
            reporte.Add($"{headers[i]},{valores.Sum()},
{valores.Average()},{valores.Max()}");
    }

    File.WriteAllLines(txtReporte.Text, reporte);
    MessageBox.Show("Reporte CSV generado.", "Éxito");
}
```

Explicación

- Analiza cada columna numérica y genera estadísticas en un nuevo CSV.

Ejercicio 45 — Comparador de JSON (Identidad de contenido y líneas)

Objetivo
Comparar dos archivos JSON y detectar diferencias en contenido.

Diseño

- `TextBox` `txtJSON1`
- `TextBox` `txtJSON2`
- `Button` `btnComparar`
- `ListBox` `lstDiferencias`

Código

```
private void btnComparar_Click(object sender, EventArgs e)
{
    string[] json1 = File.ReadAllLines(txtJSON1.Text);
    string[] json2 = File.ReadAllLines(txtJSON2.Text);
    lstDiferencias.Items.Clear();

    int max = Math.Max(json1.Length, json2.Length);
    for (int i = 0; i < max; i++)
    {
        string j1 = i < json1.Length ? json1[i] : "";
        string j2 = i < json2.Length ? json2[i] : "";
        if (j1 != j2)
            lstDiferencias.Items.Add($"Linea {i+1}: '{j1}' vs
'{j2}'");
    }
}
```

Explicación

- Compara línea por línea y muestra las diferencias.

Ejercicio 46 — Normalizador de JSON (Aplicar indentación y CamelCase)

Objetivo
Formatear JSON con indentación y convertir nombres de propiedades a CamelCase.

Diseño

- `TextBox` `txtJSON`
- `Button` `btnNormalizar`

Código

```
using System.Text.Json;
using System.Text.Json.Serialization;

private void btnNormalizar_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    var doc = JsonSerializer.Deserialize<JsonElement>(json);
    var options = new JsonSerializerOptions { WriteIndented = true, PropertyNamingPolicy = JsonNamingPolicy.CamelCase };
    File.WriteAllText(txtJSON.Text, JsonSerializer.Serialize(doc, options));
    MessageBox.Show("JSON normalizado.", "Éxito");
}
```

Explicación

- Aplica formato legible y nombres de propiedad en CamelCase.

Ejercicio 47 — Creador de CSV desde JSON Array (Inversa del Ej. 7)

Objetivo
Generar un CSV a partir de un JSON que contiene un array de objetos.

Diseño

- `TextBox` `txtJSON`
- `TextBox` `txtCSV`
- `Button` `btnConvertir`

Código

```
using System.Text.Json;


private void btnConvertir_Click(object sender, EventArgs e)
{
    string json = File.ReadAllText(txtJSON.Text);
    var lista = JsonSerializer.Deserialize<List<Dictionary<string, object>>>(json);
    var headers = lista[0].Keys.ToArray();
    List<string> lineas = new List<string> { string.Join(",", headers) };


    foreach (var item in lista)
        lineas.Add(string.Join(",", headers.Select(h => item[h].ToString())));


    File.WriteAllLines(txtCSV.Text, lineas);
    MessageBox.Show("CSV generado desde JSON Array.", "Éxito");
}
```

Explicación

- Similar al Ej. 21, crea CSV con encabezados del primer objeto.

Ejercicio 48 — Verificador de Integridad CSV (Campos vacíos y columnas inconsistentes)

Objetivo
Detectar filas con campos vacíos o columnas faltantes.

Diseño

- `TextBox` `txtCSV`
- `Button` `btnVerificar`
- `ListBox` `lstProblemas`

Código

```
private void btnVerificar_Click(object sender, EventArgs e)
{
    var lines = File.ReadAllLines(txtCSV.Text);
    int columnas = lines[0].Split(',').Length;
    lstProblemas.Items.Clear();

    for (int i = 1; i < lines.Length; i++)
    {
        var campos = lines[i].Split(',');
        if (campos.Length != columnas || campos.Any(c =>
string.IsNullOrWhiteSpace(c)))
            lstProblemas.Items.Add($"Fila {i+1} inconsistente o
con vacío");
    }
}
```

Explicación

- Revisa que todas las filas tengan el número correcto de columnas y sin campos vacíos.

Ejercicio 49 — Generador de Muestra CSV (Primeras N filas)

Objetivo
Crear un CSV con las primeras N filas de otro CSV.

Diseño

- `TextBox` `txtCSV`
- `TextBox` `txtN`
- `TextBox` `txtSalida`
- `Button` `btnMuestra`

Código

```
private void btnMuestra_Click(object sender, EventArgs e)
{
    int n = int.Parse(txtN.Text);
    var lines = File.ReadAllLines(txtCSV.Text).Take(n + 1); // +1 por encabezado
    File.WriteAllLines(txtSalida.Text, lines);
    MessageBox.Show("CSV de muestra creado.", "Éxito");
}
```

Explicación

- Permite obtener un subconjunto inicial del CSV original.

Ejercicio 50 — Administrador de Archivos CSV/JSON (Resumen de funcionalidades)

Objetivo
Aplicación que integra varias funciones: leer, escribir, filtrar, convertir y generar reportes de CSV y JSON.

Diseño

- `TextBox` para ruta
- `Buttons` para funciones: leer, filtrar, convertir, reporte
- `ListBox` o `DataGridView` para mostrar resultados

Código

```
// Ejemplo simplificado de menú de funciones
private void btnLeer_Click(object sender, EventArgs e)
{
    string ruta = txtArchivo.Text;
    if (ruta.EndsWith(".csv"))
        MessageBox.Show("Lectura CSV: " + string.Join(" | ",
File.ReadAllLines(ruta).Take(5)));
    else if (ruta.EndsWith(".json"))
        MessageBox.Show("Lectura JSON: " +
File.ReadAllText(ruta).Substring(0, Math.Min(200,
File.ReadAllText(ruta).Length)));
}


// Otras funciones pueden integrarse con botones adicionales
siguiendo los ejercicios anteriores.
```

Explicación

- Esta aplicación actúa como “hub” integrando todas las funcionalidades de los ejercicios anteriores.

Epílogo

A lo largo de estas páginas has recorrido un camino que va mucho más allá del código. Has aprendido a crear formularios, manipular eventos, diseñar interfaces, y dar vida a programas que interactúan con las personas. Pero, sobre todo, has aprendido a pensar como una programadora: a dividir un problema en partes, a construir soluciones paso a paso, y a no rendirte cuando algo no funciona a la primera.

Windows Forms es solo una puerta de entrada a un universo más amplio dentro del desarrollo de aplicaciones. Cada control, cada evento y cada propiedad que exploraste son piezas de un lenguaje que te permite expresarte con precisión y creatividad. Lo que antes era solo una ventana en blanco, ahora puede transformarse en una herramienta útil, un juego, una agenda, una idea hecha software.

Este libro no termina aquí. Es un punto de partida. El conocimiento que has adquirido te permitirá avanzar hacia proyectos más complejos: bases de datos, patrones de diseño, aplicaciones conectadas a la nube o interfaces modernas con WPF o .NET MAUI. Lo importante es seguir creando, seguir explorando y, sobre todo, mantener viva la curiosidad que te trajo hasta aquí.

Recuerda: la programación no se aprende solo leyendo o memorizando, sino practicando, imaginando y resolviendo. Cada error que enfrentes será una oportunidad para mejorar. Cada proyecto, una nueva aventura.

Gracias por llegar hasta el final.
Tu siguiente línea de código te espera.

Luis Minguillón Pascual